UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Facultad de Ciencias e Ingenierías Físicas y Formales

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

PROCESO DE RECICLAJE DE PLÁSTICO AUTOMATIZADO POR RADIOFRECUENCIA

TESIS DE GRADO PARA OPTAR EL TÍTULO DE

INGENIERO ELECTRÓNICO

Autor: Carlos Alberto Vilcherrez Arroyo

Arequipa-Perú

2016

PRESENTACION

El presente trabajo se origina del intercambio de ideas con mis compañeros de grupo en el curso de “Proyecto” y, luego, encaminado como tema de tesis conjuntamente con mi profesor Ing. Ronald Coaguila, quien me guió en la conceptualización de esta propuesta ahora desarrollada.

La iniciativa de automatizar una planta de reciclaje de plásticos postula una contribución a la eficiencia de los procesos industriales en esta línea, dado que la emisión de residuos plásticos es un problema cada vez más frecuente y demanda alternativas de solución en el contexto nacional y mundial.

El problema de degradación de los plásticos, originado por el variado uso diario del ser humano, tiene efectos nocivos en el medio ambiente y en el bienestar social; ante ello existen diversas respuestas que tienden a mitigar el problema. Las soluciones basadas en plantas recicladoras se desarrollan bajo procesos mecánicos; en esta tesis se propone una mejora de los procesos a base de la automatización inalámbrica mediante comunicación por radiofrecuencia.

Para el desarrollo de esta investigación se diseñó y construyó un *modelo de planta de reciclaje*, en el cual se hicieron las pruebas simuladas y la aplicación de los procesos automatizados. Los alcances y limitaciones de este modelo son propios de su carácter demostrativo y académico; no obstante, es una valiosa herramienta explicativa de los procesos y resultados obtenidos.

Los resultados de esta investigación demuestran que es factible la implementación de las técnicas de automatización para contribuir en tres aspectos claves de los procesos industriales: seguridad, eficiencia y calidad de las tecnologías aplicadas.

Bajo estas consideraciones, con el presente trabajo se sientan bases para profundizar investigaciones en esta línea de la actividad industrial, con el propósito de seguir contribuyendo en su mejora paulatina y, en lo posible, en la implementación a escala productiva.

Carlos A. Vilcherrez

DEDICATORIA

A la memoria de mi abuelo César Arroyo, quien con su consejo y cariño supo guiarme por el sendero de la vida

A mis padres José y Ana María, y a mi hermano José Carlos, por todo su apoyo incondicional y su cariño de familia que ayudaron a mi formación profesional; a ellos les estaré agradecido toda la vida

AGRADECIMIENTO

A mi asesor de tesis, Ing. Ronald Coaguila, por su orientación profesional y sus conocimientos impartidos y dedicados en la elaboración de este trabajo

A mis profesores, por sus enseñanzas y transferencia de conocimientos en el día a día de mi formación profesional

A mi Universidad, por su valiosa contribucion en mi formación integral, tanto académica como personal.

INDICE

PRESENTACION 2

DEDICATORIA 4

AGRADECIMIENTO 5

RESUMEN 17

CAPITULO I: PLANTEAMIENTO DEL PROBLEMA 19

CAPITULO II: JUSTIFICACIÓN DE LA INVESTIGACIÓN 22

CAPITULO III: OBJETIVOS Y ALCANCES 24

CAPITULO IV: METODOLOGÍA DE LA INVESTIGACIÓN 26

CAPITULO V: MARCO TEÓRICO Y ESQUEMA CONCEPTUAL 28

5.1. FUNDAMENTO MECÁNICO DEL PROCESO 28

5.1.1. Principios Técnicos del reciclaje 28

5.1.2. Etapas del Proceso de Planta 29

5.2. FUNDAMENTOS DE LA AUTOMATIZACIÓN DE PROCESOS 33

5.2.1. Control por PWM 34

5.2.2. Control de Temperatura por PID 38

5.3. AUTOMATIZACIÓN POR RADIOFRECUENCIA 40

CAPITULO VI: DISEÑO DEL HARDWARE Y SOFTWARE 43

6.1. IDENTIFICACIÓN DE PUNTOS A AUTOMATIZAR 43

6.2. FUNCIONAMIENTO GENERAL DEL PROCESO AUTOMATIZADO 46

6.3. DISEÑO DEL PID 49

6.4. DISEÑO DEL CIRCUITO DE CONTROL DE TEMPERATURA 62

6.4.1. Detector de cruce por cero 63

6.4.2. Control de potencia 72

6.5. DISEÑO DEL CIRCUITO PARA EL CONTROL DE MOTORES 73

6.6. MÓDULOS INALÁMBRICOS 77

6.6.1. Nodos de trabajo 79

6.6.2. Configuración de módulos XBEE 81

6.7. FUENTES DE ALIMENTACIÓN Y REGULADORES DE VOLTAJE 91

6.8. DISEÑO DE REGULADORES DE VOLTAJE 92

6.9. SENSORES ULTRASÓNICOS Y DE MOVIMIENTO 97

6.9.1. Sensores Ultrasónicos: 97

6.9.2. Sensor de movimiento 98

6.10. EL MICROCONTROLADOR ATMEGA1284P Y SOFTWARE DE AUTOMATIZACIÓN 100

6.10.1. Control de ángulo de disparo 100

6.10.2. PWM 103

6.10.3. Módulos XBEE 106

6.10.4. Sensores ultrasónicos 110

6.11. DISEÑO DEL SOFTWARE 114

CAPITULO VII: DESARROLLO DEL PROYECTO 120

7.1. COMPONENTES UTILIZADOS 120

7.1.1. Componentes Electrónicos 120

7.1.2. Sensores. 121

7.1.3. Componentes Mecánico-Eléctricos 122

7.2. CIRCUITOS Y DISPOSITIVOS APLICADOS A LA AUTOMATIZACIÓN 122

7.2.1. Circuito de Placa Maestra 122

7.3. PLACA DE MANDO 145

CAPITULO VIII: PRUEBAS DE FUNCIONAMIENTO 150

8.1. PRUEBAS EN EL PID PARA LA CALIBRACIÓN 150

8.2. PRUEBAS DEL PID UNA VEZ CALIBRADO A DIFERENTES TEMPERATURAS 153

8.3. PRUEBA DEL PROCESO AUTOMATIZADO POR RADIOFRECUENCIA GENERAL 157

CAPITULO IX: RESULTADOS DE LA INVESTIGACIÓN 164

CAPITULO X: COMPARACIÓN CON OTROS PROCESOS **171**

10.1. COMPARACION CON PROCESO MECÁNICO **171**

10.2. COMPARACIÓN CON PLC **177**

CAPITULO XI: CONCLUSIONES **179**

CAPITULO XII: MEJORAS O RECOMENDACIONES **181**

CAPÍTULO XIII: REFERENCIAS BIBLIOGRÁFICAS **182**

Páginas Visitadas **182**

ANEXOS **184**

ANEXO 1.- COMPORTAMIENTO DEL CALEFACTOR **185**

ANEXO 2.- .- PROGRAMACIÓN HECHA EN C DEL PROCESO **188**

ANEXO 3.- DISEÑO DE PISTAS Y PLACA **242**

INDICE DE FIGURAS

CAPÍTULO V: MARCO TEÓRICO Y ESQUEMA CONCEPTUAL

FIGURA 5.1. Diagrama de flujo del reciclado de plástico

FIGURA 5.2. Gráfica del funcionamiento del PWM según el ancho de pulso

FIGURA 5.3. Ciclo de trabajo de 25%

FIGURA 5.4. Ciclo de trabajo de 50%

FIGURA 5.5. Ciclo de trabajo de 75%

FIGURA 5.6. Diagrama de bloques del PID

FIGURA 5.7. Espectro de frecuencias

CAPÍTULO VI: DISEÑO DEL HARDWARE Y SOFTWARE

FIGURA 6.1. Modelo a escala del proceso a automatizar

FIGURA 6.2. Diagrama de flujo general de placa maestra

FIGURA 6.3. Diagrama de flujo general de control de mando

FIGURA 6.4. Proceso de extrusión

FIGURA 6.5. Diagramaa de bloques general

FIGURA 6.6. Prueba para el comportamiento del calefactor

FIGURA 6.7. Método trapezoidal para el PID discreto

FIGURA 6.8. Esquemático del control de temperatura

FIGURA 6.9. Esquemático del circuito de cruce por cero

FIGURA 6.10. Señal antes y después del rectificador de onda

FIGURA 6.11. Rd en el circuito del cruce por cero

FIGURA 6.12. Optoacoplador en el circuito de cruce por cero

FIGURA 6.13. Señal rectificada después del puente de diodos

FIGURA 6.14. Señal del cruce por cero para poder ser reconocida

FIGURA 6.15. Esquemático del control del disparo del triac

FIGURA 6.16. Esquemático obtenido del dathaseet del moc3021

FIGURA 6.17. Motor para el control de faja transportadora

FIGURA 6.18. Motor para el control de extrusión

FIGURA 6.19. Esquemático del circuito del control de motores para el diseño de Rm

FIGURA 6.20. Esquemático del circuito del control de motores para el diseño de Rin

FIGURA 6.21. Módulo XBEE en el transmisor

FIGURA 6.22. Módulo XBEE en el receptor

FIGURA 6.23. Esquemático del módulo XBEE

FIGURA 6.24. Interfaz de configuración inicial para módulos XBEE del progama X-CTU

FIGURA 6.25. Configuración de XBEE en modo router

FIGURA 6.26. Configuración de parámetros en el XBEE

FIGURA 6.27. Configuración de baudrate en el XBEE

FIGURA 6.28. Configuración del canal de operación en el XBEE

FIGURA 6.29. Combinaciones de 8 bits en código HEX en el módulo XBEE.

FIGURA 6.30. Pruebas de envío de datos entre módulos XBEE

FIGURA 6.31. Pantalla LCD en el control de mando para visualización de parámetros

FIGURA 6.32. Fuente variable de 0 a 15v

FIGURA 6.33. Fuente variable de 0 a 24V

FIGURA 6.34. Esquemático del regulador de voltaje de 15V

FIGURA 6.35. Esquemático del regulador de voltaje de 5V

FIGURA 6.36. Esquemático del regulador de voltaje de 3.3V en el control de mando

FIGURA 6.37. Esquemático del diseño del regulador de voltaje en el datasheet del LM2576

FIGURA 6.38. Trasientes generadas por los motores

FIGURA 6.39. Sensor ultrasónico HC-SR04

FIGURA 6.40. Sensores ultrasónicos en Tolvas principal y secundaria respectivamente

FIGURA 6.41. Sensor PIR de movimiento

FIGURA 6.42. Esquemático del sensor de movimiento PIR

FIGURA 6.43. Control de disparo del triac

FIGURA 6.44. Cruce por cero

FIGURA 6.45. Esquemático del control de temperatura con el microcontrolador

FIGURA 6.46. Variación del PWM a 8.80V.

FIGURA 6.47. Variación del PWM a 6V

FIGURA 6.48. Variación del PWM a 9.60V

FIGURA 6.49. Tabla de pines de los módulos XBEE

FIGURA 6.50. Esquemático del módulo XBEE

FIGURA 6.51. Esquemático de los pines utilizados en el módulo XBEE

FIGURA 6.52. Esquemático del microcontrolador con referencia al regulador de 3.3V utilizado

FIGURA 6.53. Adaptador lógico en el microcontrolador para los sensores ultrasónicos

FIGURA 6.54. Esquemático del adaptador lógico para hallar Rb

FIGURA 6.55. Aparición de tierra virtual

FIGURA 6.56. Diagrama de control general

FIGURA 6.57. Interrupción de sensores ultrasónicos

FIGURA 6.58. Interrupción de cruce por cero

FIGURA 6.59. Interrupción medio grado

FIGURA 6.60. Digrama de flujo del control de mando

FIGURA 6.61. Interrupcion de pulsadores

CAPÍTULO VII: DESARROLLO DEL PROYECTO

FIGURA 7.1. Circuito General en placa

FIGURA 7.2. Integrado LM2576

FIGURA 7.3. Integrado LM1117

FIGURA 7.4. Esquema circuital y diagrama interno del circuito integrado LM2576

FIGURA 7.5. Circuito en placa de los reguladores de voltaje

FIGURA 7.6. Drivers de disparo IR2110

FIGURA 7.7. Diagrama esquemático del circuito integrado IR2110

FIGURA 7.8. Circuito en placa para el control de motores

FIGURA 7.9. Circuito en placa del control de temperatura

FIGURA 7.10. Circuito en placa para el control de temperatura

FIGURA 7.11. Circuito en placa del control de cruce por cero

FIGURA 7.12. Circuito en placa del control de disparo del triac

FIGURA 7.13. Chip Max31855 para sensor de temperatura

FIGURA 7.14. Diagrama del chip sensor de temperatura

FIGURA 7.15. Circuito en placa para el chip MAX31855

FIGURA 7.16. Micocontrolador atmega modelo 1284P

FIGURA 7.17. Diagrama de los pines del microcontrolador 1284P

FIGURA 7.18. diagrama de arreglo interno del micocontrolador atmega modelo 1284P

FIGURA 7.19. Variación de velocidad en relación al ancho de pulso

FIGURA 7.20. Circuito en placa del atmega 1284P

FIGURA 7.21. Placa de Mando Cara Superior

FIGURA 7.22. Placa de Mando Cara Inferior

FIGURA 7.23. Esquemático del circuito reguladore de voltaje

Figura 7.24. Circuito en placa de divisor de voltaje en el control de mando

FIGURA 7.25. Esquemático del circuito del microcontrolador 16A

FIGURA 7.26. Circuito en placa del microcontrolador 16A

CAPÍTULO VIII: PRUEBAS DE FUNCIONAMIENTO

FIGURA 8.1. Prueba del funcionamiento del PID

FIGURA 8.2. Prueba de conexión de módulos RF

FIGURA 8.3. Material de plástico picado para las pruebas correspondientes

FIGURA 8.4. Ingreso de material al proceso

FIGURA 8.5. Medición del llenado de tolva principal

FIGURA 8.6. Variación del motor por PWM mediante pulsos

FIGURA 8.7. Medición de llenado de tolva secundaria

FIGURA 8.8. Variación de temperatura en la pantalla LCD

FIGURA 8.9. Elección de la temperatura a la que se quiere trabajar

FIGURA 8.10. Activación de motor del extrusor

CAPÍTULO IX: RESULTADOS DE LA INVESTIGACIÓN

FIGURA 9.1. Producto final en el proceso automatizado

INDICE DE TABLAS

CAPÍTULO VI: DISEÑO DEL HARDWARE Y SOFTWARE

TABLA 6.1. Variación de temperatura en el tiempo del calefactor

TABLA 6.2. Constante Kp y Tiempos (Ti y Td) del PID

TABLA 6.3. Constantes del PID

TABLA 6.4. Constantes halladas del PID

CAPÍTULO VII: DESARROLLO DEL PROYECTO

TABLA 7.1. Tipos y Características de los Sensores de Temperatura

CAPÍTULO VIII: PRUEBAS DEL FUNCIONAMIENTO

TABLA 8.1. Prueba del PID en el calefactor para estabilizar a 150°c

TABLA 8.2. Prueba del PID en el calefactor para estabilizar a 160°c

TABLA 8.3. Prueba del PID en el calefactor para estabilizar a 170°c

TABLA 8.4. Prueba General del PID para 150°c

TABLA 8.5. Prueba del PID calibrado a 40°c

TABLA 8.6. Pruebas del PID calibrado a 300°c

CAPÍTULO IX: RESULTADOS DE LA INVESTIGACIÓN

TABLA 9.1. Variables monitoreadas y controladas en el proceso de reciclaje de plásticos

INDICE DE GRÁFICAS

CAPÍTULO VI: DISEÑO DEL HARDWARE Y SOFTWARE

GRÁFICA 6.1. Comportamiento del calefactor en el extrusor

GRÁFICA 6.2. Modelo para hallar las constantes del PID

GRÁFICO 6.3. Relación entre Id e If para diseño

CAPÍTULO VIII: PRUEBAS DEL FUNCIONAMIENTO

GRÁFICA 8.1. prueba con PID calibra a 40° c

GRÁFICA 8.2. prueba con PID calibra a 300° c

CAPÍTULO IX: RESULTADOS DE LA INVESTIGACIÓN

GRÁFICA 9.1.Prueba PID a 40°c

GRÁFICA 9.2.Prueba PID a 300°c

RESUMEN

En el presente trabajo de tesis se sustenta el diseño, modelamiento e implementación de una nueva tecnología de automatización para el control y monitoreo de procesos en reciclaje de plástico, constituyendo una innovación tecnológica de los procesos industriales, cuyo objetivo es desarrollar una interfaz nueva, fácil y sencilla; además, es segura para el usuario gracias a la radiofrecuencia utilizada como medio de comunicación.

Para demostrar el sistema automatizado de planta se detalla el análisis, diseño y desarrollo del sistema propuesto, utilizando como instrumento de demostración experimental un *módulo de planta de reciclaje*, obteniendo así un modelamiento óptimo, eficaz y eficiente según los requerimientos necesarios en las diversas fases de producción, concluyendo que es totalmente factible implementarlo a escala industrial.

El modelo demostrativo utilizado, a pequeña escala, permite la implementación del sistema de control tomando datos obtenidos mediante sensores de movimiento, con los cuales se detecta el ingreso de materia prima al proceso, en este caso plástico (PET); se complementa con sensores ultrasónicos, los cuales miden la altura del llenado de las tolvas tanto la principal como la secundaria; además, con la instalación de PWM (modulación por ancho de pulso) se optimiza el funcionamiento del motor de la faja transportadora. Estos datos obtenidos de los sensores son interpretados por un circuito electrónico (placa maestra), la cual envia los datos mediante señales de comunicación por radiofrecuencia (RF) a un control de mando; este control de mando, de igual

manera, interpreta las señales y las muestra en una pantalla LCD en forma de un menú fácil y sencillo de usar para que, posteriormente, el usuario controle el proceso mediante una interfaz de pulsadores en el control de mando, teniendo dos variables controladas: temperatura y velocidad de motores. Una vez modificada la variable, esta señal es interpretada nuevamente por los circuitos electrónicos del control de mando y placa maestra, y enviada nuevamente por radiofrecuencia para ser modificada en el proceso. El sistema se completa con dos parámetros monitoreados: ingreso de material y altura en el llenado de tolvas, los cuales se visualizan en el LCD.

Así mismo, en el desarrollo del proyecto, se han diseñado y construido los circuitos hechos en placa y sus componentes utiizados. Sobre esta base fue que se realizaron las pruebas correspondientes para el óptimo desempeño del proceso.

Los resultados obtenidos de las pruebas realizadas, permiten concluir que se cumplió satisfactoriamente con los objetivos trazados al inicio de esta investigación.

CAPITULO I: PLANTEAMIENTO DEL PROBLEMA

La automatización en nuestro país se va desarrollando cada vez más y más, sin embargo aún existen muchas más tecnologías e ideas por desarrollar.

La automatización es la tecnología que se le aplica a un proceso de planta, para poder hacer una producción más eficiente y mejorar la calidad del producto; aunque en nuestro país encontramos esta automatización en una forma escasa y poco eficiente, esta existe; sin embargo, aún existe un gran abismo entre nuestra tecnología de automatización en comparación a la de otros países.

Además, si la automatización en nuestro país como se mencionó anteriormente es poca a comparación de otros países, es menos probable poder encontrar una automatización inalámbrica, es decir, poder variar parámetros y monitorear todo el proceso en general inalámbricamente.

En este proyecto lo que se busca es poder automatizar un proceso de reciclaje de plástico, ya que en nuestro país si bien existen unas pocas plantas de reciclaje, pues estas carecen de una automatización completa, y no con el proceso que se muestra en este trabajo.

En este proceso de planta el cual según el modelo a escala presentado cuenta con una tolva principal, una faja transportadora, una tolva secundaria y un extrusor, el problema se centra en la automatización de estos 4 puntos

específicamente, para poder hacer un proceso seguro, eficaz y eficiente; y los presentaremos a continuación:

- Medir de alguna forma y poder monitorear el llenado en la tolva principal, porque al haber cualquier problema en el proceso siguiente, la tolva principal se llenaría hasta que el material (plástico) desborde, lo cual genera pérdidas en todo sentido. Además, si en el proceso no está ingresando material pues sería ineficiente que este siga funcionando y generaría gasto de maquinaria, pérdida de energía y costos.

- Transportar material de manera eficiente por si el material llega a retrasarse por cualquier motivo, en segundo lugar, el tipo de material no siempre será el mismo por lo que para algún tipo de material será necesario tener un poco más de potencia en el motor de la faja, y en tercer lugar por algún motivo que la tolva se esté llenando con demasiado material pues la faja necesitará trabajar a su mayor velocidad para descongestionar dicha tolva. Para lo que un motor fijo ya sea a su máxima, media o baja potencia no servirá.

- Medir y monitorear el llenado en la tolva, ya que de igual manera que la anterior tolva, la tolva secundaria se llenaría hasta que el material (plástico) desborde, y esto genera pérdidas al momento de hacer mantenimiento y reparaciones no solo a la maquinaria dañada si no que al desbordar el material generaría daños a otras máquinas y en tema de seguridad este problema sería muy grave.

- En el extrusor, al ser este el punto más importante del proceso, tienes varios problemas que resolver, primeramente, debido al trabajo con altas temperaturas de calor en el extrusor, por seguridad debe ser una prioridad que el personal quien controla y monitorea no se acerque a esta área de trabajo. El problema específicamente aquí es poder mantener la temperatura constante, aunque esta sufra perturbaciones, y aunque esto pueda lograrse, ¿que pasaría al cambiar de material por uno tal vez más grueso y resistente u otro más liviano y menos resistente?, entonces ¿Cómo variar la temperatura?, claro se puede apagar el extrusor y esperar a que este enfríe, lo cual generará pérdida de tiempo y de ganancias, además si se quiere variar dicha temperatura en caliente (extrusor prendido), el riesgo de la persona que lo haga es alto.

Finalmente, la automatización en los procesos de planta tiene que ser lo más específico y óptimo posible, y un plus más allá de ello es hacerlo inalámbricamente; en nuestro país, existen muchos o la mayoría de los procesos automatizados con una sola secuencia de procesos y un bucle que constantemente está comparando esta secuencia general de procesos; sin embargo, hay ocasiones, ya sean por accidente, incidentes o por cualquier otra razón, estas secuencias programadas no puedan ser visualizadas ni monitoreadas, y por lo tanto no pueden ser reparadas al instante, lo cual genera pérdidas económicas.

CAPITULO II: JUSTIFICACIÓN DE LA INVESTIGACIÓN

La presente investigación propone introducir un ***cambio tecnológico*** en la automatización en un proceso de reciclaje de plástico a fin de contribuir a la mitigación de los efectos ambientales y reducir los efectos nocivos en el bienestar social, derivando en incrementos de los beneficios económicos de las entidades dedicados a esta actividad ambiental o giro de negocio empresarial.

El cambio tecnológico se sustenta en implementar un proceso automatizado en una planta de reciclaje de plásticos, incorporando la comunicación inalámbrica por radiofrecuencia con el objeto de mejorar determinadas fases del proceso productivo, identificando los puntos críticos o los elementos relevantes en los cuales se apliquen automatización inalámbricamente a la gestión productiva. Con este mecanismo se espera que los resultados derivados de la automatización tengan significación en la gestión de la entidad responsable del reciclaje, sea pública o privada, principalmente en los aspectos de seguridad, calidad de producto, control ambiental y **beneficios económicos**.

La esencia de la investigación se sustenta en comparar el funcionamiento mecánico con el automatizado y más aún con otros tipos de automatizaciones como PLCs u otros, en la producción de la planta. El proceso mecánico es convencional y es la base de comparación. La automatización se sustenta en el uso de sensores con parámetros medibles y variables a fin de evaluar si el proceso está siendo óptimo; cada parámetro de medición se visualiza en un tablero de control y es precisamente aquí donde se aplica la automatización mediante comunicación por radiofrecuencia para enviar y recibir la información

del proceso de producción, la cual constituye un valioso medio de conocimiento de la eficiencia y, eventualmente, para adoptar acciones correctivas que puedan tomarse desde el tablero de control y en algunos casos, por motivos de seguridad, desde la planta misma.

CAPITULO III: OBJETIVOS Y ALCANCES

Debido a que la automatización en plantas industriales, en nuestro país aún no está desarrollada con las tecnologías más avanzadas como en los países más desarrollados, el principal objetivo de este proyecto es automatizar una planta en este caso de reciclaje de plástico con una tecnología más avanzada a las ya conocidas y con el plus de ser inalámbrica por radiofrecuencia.

En un punto más detallado de la automatización, como se mencionó en el planteamiento del problema, el modelo a escala consta de una tolva principal, una faja de transporte, una tolva secundaria y un extrusor, para estos puntos críticos, los objetivos son los siguientes:

- Medir y monitorear el llenado de la tolva principal desde un control de mando inalámbrico por radiofrecuencia, mediante un sensor ultrasónico, además de contar con un sensor de movimiento para cuando el material deje de ingresar en el proceso, la planta no funcione ineficientemente al no haber material procesable en ella.

- Variar la velocidad del motor de la faja de transporte mediante PWM en un rango establecido de velocidad, desde el control de mando inalámbrico por radiofrecuencia.

- Medir y monitorear el llenado de la tolva secundaria desde un control de mando inalámbrico por radiofrecuencia, mediante un sensor ultrasónico.

- Medir y variar la temperatura del calefactor en el extrusor mediante PID desde el control inalámbrico, además de trabajar el funcionamiento

mecánico (motor) de una manera aislada, esto quiere decir que el motor del extrusor funcionará desde el control de mando inalámbrico totalmente aparte del control de temperatura.

Otro objetivo importante es presentar una opción de control y monitoreo de proceso que sea seguro para el usuario.

Además, otro objetivo también importante es poder contribuir al cuidado del medio ambiente, en este caso reciclando botellas de plástico (PET), y poder generar un cambio y un impacto ambiental, y por último pero no menos importante, ofrecer una tecnología diferente, eficaz y eficiente a un costo menor al de otras formas de automatización, y de una productividad óptima por ser automatizada en cada punto crítico del proceso.

CAPITULO IV: METODOLOGÍA DE LA INVESTIGACIÓN

Esta investigación se sustenta en el análisis científico, de tipo experimental, basado en la observación - pruebas – comprobación de eventos o sucesos previamente diseñados para que ocurran, los cuales sustentan la experimentación.

Las aplicaciones y pruebas que corresponden al análisis de los procesos se hacen en el Modelo Demostrativo de Planta, partiendo de la puesta en funcionamiento del proceso mecánico. El proceso automatizado se basa en la determinación de parámetros y el control de variables para comprobar razonablemente los cambios tecnológicos por la incorporación de la automatización mediante radiofrecuencia.

Para el diseño y aplicación de los procesos se han revisado estudios e investigaciones sobre la automatización [1] y en este caso del reciclaje de plásticos en general, los cuales constituyen antecedentes útiles y una guía para el desarrollo del estudio y, además, permiten hacer comparaciones de cómo se plantean proyectos para resolver este problema.

También se ha procedido a realizar consultas especializadas de profesionales conocedores de los temas de reciclaje y automatización, a fin de capitalizar de su experiencia y sus conocimientos en torno a esta investigación.

[1] Plan de Manejo Ambiental de Residuos Sólidos del Distrito José Luis Bustamante Y Rivero, Municipalidad Distrital de JL Bustamante y Rivero, 2015

Ver reportaje en https://www.youtube.com/watch?v=8pRkk3B8n8g

Las pruebas en el Modelo Demostrativo de Planta se han realizado por aproximaciones sucesivas; es decir, aplicando el método de ensayo – error, hasta lograr que la simulación del funcionamiento de planta permita determinar los parámetros y el control de variables en los que se sustenta esta investigación.

En cada aplicación piloto de las pruebas se han registrado los datos observados y se ha comprobado su validez para medir los cambios provocados por la automatización.

La información de las aplicaciones y controles se ha ordenado en dos programas informáticos denominados: "*Programa placa maestra"* y *"Programa control de mando"*, los cuales han sido construidos en lenguaje C. Estos programas permiten la simulación del proceso automatizado en sus componentes básicos de emisión – transmisión – recepción de datos.

CAPITULO V: MARCO TEÓRICO Y ESQUEMA CONCEPTUAL

5.1. FUNDAMENTO MECÁNICO DEL PROCESO

5.1.1. Principios Técnicos del reciclaje

El reciclaje de los plásticos significa la recuperación y el reprocesamiento de los mismos, cuando su vida útil terminó, para usarlos en nuevas aplicaciones.

El reciclado mecánico consiste en el tratamiento de los residuos plásticos por medio de la presión y el calor para volver a darles forma y conseguir otros objetos iguales o distintos de los iniciales. Por ello sólo se aplica a los termoplásticos, ya que estos materiales son reciclables por naturaleza.

Aunque el proceso de reciclado (fusión y solidificación) puede repetirse varias veces, cada vez que se lleva a cabo, el plástico tiende a perder entre el 5 y 10% de sus propiedades mecánicas, tales como elongación, tenacidad y resistencia al impacto. Por esta razón, deben restituirse estas propiedades con ayuda de aditivos, como modificadores de impacto, estabilizadores al calor, absorbedores de luz ultravioleta y cargas.

Las diferentes etapas del proceso pueden variar según la tecnología que se use. La mayoría de los equipos son similares a los empleados

normalmente en la manufactura de plásticos a partir de materia prima virgen.

5.1.2. Etapas del Proceso de Planta

Las etapas que se describen a continuación son típicas de una empresa recicladora de plástico. En la siguiente figura se presenta un diagrama de flujo del proceso de reciclado en general:

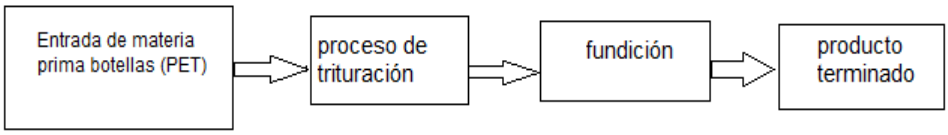


FIGURA 5.1. Diagrama de flujo del reciclado de plástico

Para entender mejor el proceso mecánico, explicaremos cada punto del proceso desde la clasificación del material hasta el producto final:

1. **Clasificación**: El reciclado mecánico se inicia con la clasificación de los residuos. Debido a la incompatibilidad de los plásticos y su dificultad para separarlos, esta etapa del proceso es fundamental. Se han desarrollado técnicas avanzadas para mejorar la clasificación ya que de ello depende la calidad del producto final. Por ejemplo: pequeñas cantidades de PVC reducen de manera muy significativa el valor comercial del PET.

   La presencia de plásticos coloreados puede significar otra dificultad, aun tratándose de plásticos de la misma resina.

El proceso de clasificación puede llevarse a cabo en forma manual o mecanizado:

- **Clasificación Manual:** Los recicladores, gracias a la experiencia del trabajo, han adquirido una gran habilidad para seleccionar y clasificar los residuos plásticos aunque desconocen la codificación internacional. Es indispensable contar con los implementos adecuados para protegerse de las sustancias tóxicas a las que se exponen.
- **Clasificación Mecanizada:** Existen en países industrializados, procesos para clasificar los residuos automáticamente, mediante bandas transportadoras y dispositivos con sensores térmicos, espectroscopios infrarrojos, entre otros. La técnica basada en espectroscopia IR (infrarrojos), es muy efectiva y ha alcanzado escala comercial. Un analizador identifica la naturaleza de la resina por medio de la luz IR reflejada en el material y transmite la orden para la retirada mecánica o neumática del objeto al contenedor correspondiente.

2. **Limpieza:** Se realiza en forma manual, escogiendo los envases de PET vacíos, su objetivo es eliminar los residuos líquidos existentes en algunos envases usados; las botellas se lavan. En este proceso son necesarias grandes cantidades de agua y detergente.

3. **Trituración:** Los plásticos son reducidos de tamaño en un molino de cuchillas, obteniendo hojuelas de plástico conocidas como scraps.

   La trituradora de cuchillas: Está diseñada específicamente para soportar una gran carga de materiales para su triturado. El cuerpo de la maquinaria está fabricado en hierro y acero reforzado. Su motor, con cuchillas de acero, se encuentra protegido contra partículas de polvo, puede triturar toda clase de materiales de plástico. Las cuchillas de esta trituradora, funcionan de manera similar a tijeras haciendo simple el triturado, sin la necesidad de pulverizar ni de utilizar calor. Las dos bocas de alimentación de material, se encuentran convenientemente ubicadas para una fácil disposición del material a pulverizar.

4. **Extrusión y Moldeo:** Es el proceso por el que se obtienen los pellets mediante una operación de extrusión.

   Se fluidiza el scrap o aglomerado utilizando un tornillo de extrusión, que en términos simplificados es un tornillo sinfín dentro de un cilindro largo. Los scraps o el aglomerado se colocan en la extrusora en el extremo del tornillo con el diámetro más grande y se comprimen mientras se lleva hacia la boquilla de extrusión. Previamente y de ser necesario se añaden los aditivos. El calor combinado de la fricción producida por el flujo y de las bandas de calefacción suplementarias provoca la fundición de la resina, extrayéndose de la mezcla los contaminantes volátiles.

La temperatura debe ser constante en cada tramo del extrusor, para lo cual se calienta con una resistencia eléctrica y se mantiene la temperatura necesaria con un sistema de refrigeración (serpentín de agua o aire).

El material plástico reciclado (pellets, scraps, o aglomerado) puede ser procesado directamente y obtener productos con menores especificaciones que las fabricadas con plástico virgen. Con la formulación adecuada, puede procesarse junto con resina virgen; esto permite una disminución en los costos de producción. También puede coextruirse entre material virgen, por ejemplo en la fabricación de botellas donde la capa intermedia es de material reciclado y, tanto la capa interna como la externa son de material virgen.

Durante el proceso de extrusión pueden usarse aditivos para cambiar el índice de fusión o el color. Los procesadores y fabricantes intentan minimizar la “historia calorífica” (una medida del número de veces que se ha fundido la resina o se ha llegado a la temperatura máxima) porque cada calentamiento degrada la resina.

Existen varias técnicas para conformar los polímeros. En su mayoría son utilizadas para aquéllos de naturaleza termoplástica. El termoplástico es calentado a una temperatura cercana o superior a la temperatura de fusión, de tal manera que se haga

plástico o líquido. Entonces es vaciado o inyectado en un molde para producir la forma deseada.

5.2. FUNDAMENTOS DE LA AUTOMATIZACIÓN DE PROCESOS

La automatización es la aplicación de procedimientos automáticos en un proceso o en una industria. La manera más conocida de automatización actual es mediante la aplicación de los denominados ***PLC* (*programable logic controller*)**, desarrollado inicialmente por la General Motors (1969) para remplazar los sistemas inflexibles cableados y ha ido evolucionando en la medida de la aparición de microprocesadores de mayor capacidad; es así que en los años 90 los microprocesadores de 32 bits abrieron la posibilidad de fábricas completamente automatizadas y con comunicación en "tiempo real".

Un PLC es un microprocesador, es decir un computador, especialmente diseñado para automatización industrial; tiene una Unidad Central de Procesamiento más conocida como CPU, interfaces de comunicación, y puertos de salida y entrada de tipo digital o análogo. Estas son solo algunas de sus características más sobresalientes. La diferencia con un computador tradicional es que no tiene teclado, pantalla ni ratón, tampoco tiene disco duro, pero sí dispone de hardware y software. La principal diferencia de un PLC con una PC es que el PLC contiene múltiples canales para medir distintas señales provenientes de sensores instalados en las maquinas o procesos que controlan.

En esencia, un PLC es un dispositivo electrónico que puede ser programado por el usuario y se utiliza en la industria para resolver problemas de secuencias en la maquinaria o procesos, ahorrando costos en mantenimiento y aumentando la confiabilidad de los equipos. Es importante conocer sus generalidades y lo que un PLC puede hacer por el proceso productivo, pues evita costos en mantenimiento y reparaciones.

En la actualidad el campo de aplicación de un PLC es muy extenso. Se utiliza fundamentalmente en procesos de maniobras de máquinas, control, señalización, etc. La aplicación de un PLC abarca procesos industriales de cualquier tipo y ofrecen conexión a red; esto permite tener comunicado un PLC con una PC y otros dispositivos al mismo tiempo, permitiendo hacer monitoreo, estadísticas y reportes.

Dos de los puntos principales en esta automatización son el control de motores por PWM y el control de temperatura por PID, los cuales también serán desarrollados a continuación.

5.2.1. Control por PWM

Para poder controlar la potencia destinada al motor DC, se va a utilizar la modulación por ancho de pulsos conocida como PWM (pulse-with modulation), el PWM de una señal o fuente de energía, consiste en una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal, o una cuadrada como en este caso), ya sea para transmitir información a través de un canal de

comunicaciones, o bien, para controlar la cantidad de energía que se envía a una carga (motor DC como en este caso).

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

Siendo:

D: el ciclo de trabajo.

τ: el tiempo en que la función es positiva (ancho del pulso).

T: el período de la función.

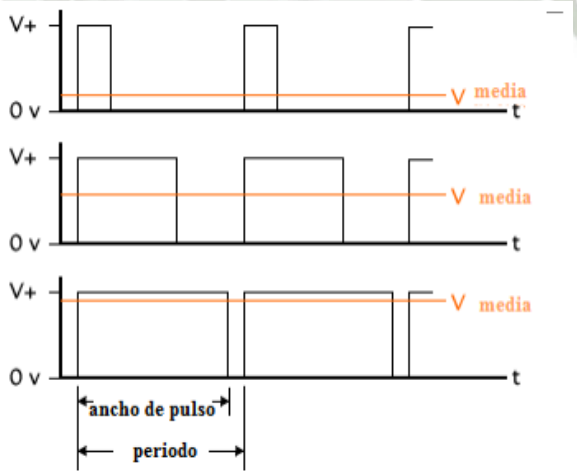


FIGURA 5.2. Gráfica del funcionamiento del PWM según el ancho de pulso[2]

[2] Gráfica del PWM bajada de internet de la siguiente dirección: http://blog-j.marcano.net.ve/index.php/2010/09/24/puente-h-con-l298-motores-dc-y-algo-de-pwm/

Las señales PWM son utilizadas comúnmente en el control de aplicaciones. Su uso principal es el control de motores de corriente continua, aunque también pueden ser utilizadas para controlar válvulas, bombas, sistemas hidráulicos, y algunos otros dispositivos mecánicos. La frecuencia a la cual la señal de PWM se generará, dependerá de la aplicación y del tiempo de respuesta del sistema que está siendo controlado. A continuación se muestran algunas aplicaciones y sus respectivas frecuencias:

- Calentar elementos o sistemas con tiempos de respuesta lentos: 10-100 Hz o superior.
- Motores eléctricos de corriente continua: 5-10 kHz o superior.
- Fuentes de poder o amplificadores de audio: 20-200 kHz o superior.

Nota: Ciertos sistemas pueden requerir frecuencias superiores a las mostradas anteriormente y dependerá del tipo de respuesta requerido.

A continuación se muestran algunos gráficos demostrando señales PWM con diferentes ciclos de trabajo:

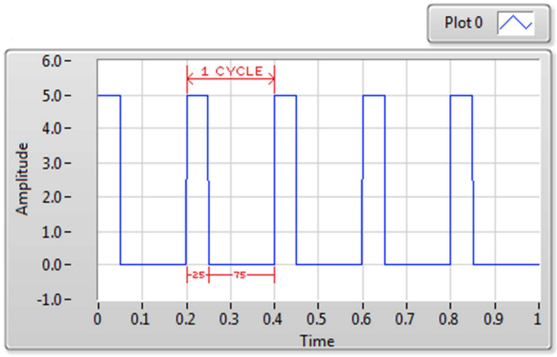


FIGURA 5.3. Ciclo de trabajo de 25%

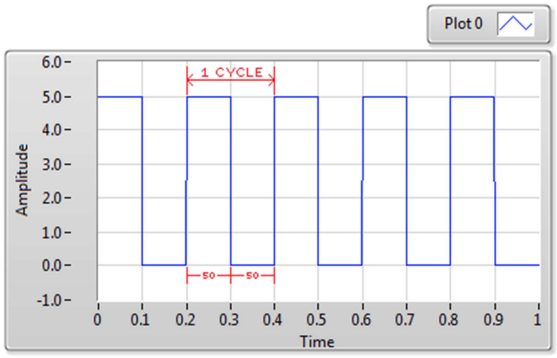


FIGURA 5.4. Ciclo de trabajo del 50%

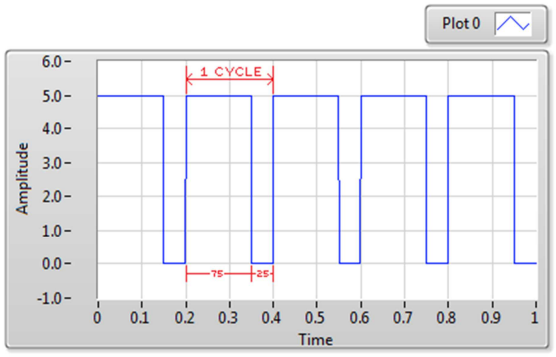


FIGURA 5.5. Ciclo de trabajo de 75%

[3] Gráficas de ejemplos de PWM descargadas de la siguiente dirección:
http://digital.ni.com/public.nsf/allkb/AA1BDEA4AA224E3E86257CE400707527

5.2.2. Control de Temperatura por PID

El control PID es un mecanismo de control que a través de un lazo de retroalimentación permite regular la velocidad, temperatura, presión y flujo entre otras variables de un proceso en general. El controlador PID calcula la diferencia entre nuestra variable real contra la variable deseada.

Consideremos un lazo de control de una entrada y una salida (SISO) de un grado de libertad:

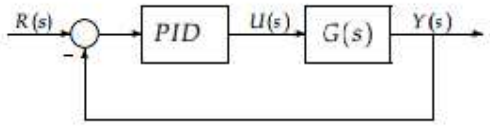


FIGURA 5.6. Diagrama de bloques del PID[4]

Los miembros de la familia de controladores PID, incluyen tres acciones: proporcional (P), integral (I) y derivativa (D). Estos controladores son los denominados P, I, PI, PD y PID.

- P: acción de control proporcional, da una salida del controlador que es proporcional al error, es decir: u(t) = KP.e(t),que descripta desde su función transferencia queda:

$$C_p(s) = K_p$$

[4] Diagrama bajado de internet del tema "Controladores PID" de la siguiente dirección: http://www.eng.newcastle.edu.au/~jhb519/teaching/caut1/Apuntes/PID.pdf

Donde Kp es una ganancia proporcional ajustable. Un controlador proporcional puede controlar cualquier planta estable, pero posee desempeño limitado y error en régimen permanente (off-set).

- I: acción de control integral: da una salida del controlador que es proporcional al error acumulado, lo que implica que es un modo de controlar lento.

$$u(t) = k_i \int_0^t e(\tau)d\tau \qquad C_i(s) = \frac{K_i}{s}$$

La señal de control u(t) tiene un valor diferente de cero cuando la señal de error e(t) es cero. Por lo que se concluye que dada una referencia constante, o perturbaciones, el error en régimen permanente es cero.

- PID: acción de control proporcional-integral-derivativa, esta acción combinada reúne las ventajas de cada una de las tres acciones de control individuales. La ecuación´ de un controlador con esta acción combinada se obtiene mediante:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(\tau)d\tau + K_p T_d \frac{de(t)}{dt}$$

Y su función de transferencia es:

$$\mathrm{C_{PID}}(\boldsymbol{s}) = \mathrm{K_p}(1 + \frac{1}{T_i s} + \mathrm{T_d}\boldsymbol{s})$$

Nota: Existen también la acción de control PI y PD, pero en este caso no los estamos mencionando.

5.3. AUTOMATIZACIÓN POR RADIOFRECUENCIA

La comunicación es un factor esencial en el desarrollo económico y social del ser humano. En términos sencillos, es la trasmisión de información de un lugar a otro. En términos tecnológicos, la comunicación es la transmisión de información desde un sistema emisor mediante un canal de comunicación para transmitir el mensaje y un sistema receptor. El canal de comunicación es el **medio** por el cual se trasmite información mediante señales (perturbaciones del medio) que se originan en el sistema emisor y llegan hasta el sistema receptor. Las señales pueden ser analógicas o digitales

Para la parte de la comunicación, la cual será inalámbricamente, ésta será por radiofrecuencia. La señal por radiofrecuencia tiene lugar cuando una señal, en el rango de 30KHz a 300GHz, se propaga de transmisor a receptor. En el siguiente gráfico podemos apreciar el espectro de frecuencias y ver en qué rango se encuentra la comunicación que utilizaremos. (ver figura 5.7.).

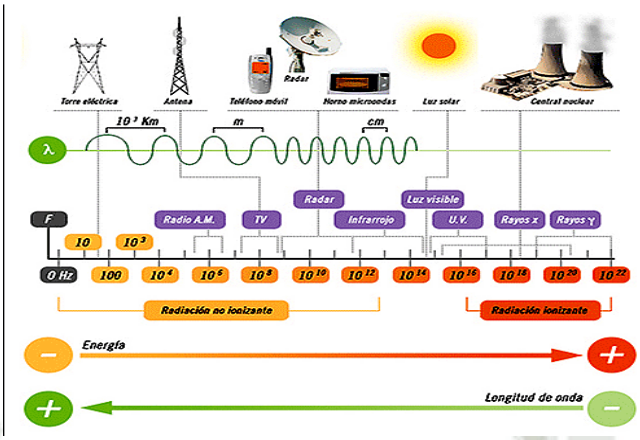


FIGURA 5.7. Espectro de frecuencias[5]

La comunicación por radiofrecuencia (RF) es básicamente una emisión electromagnética que se produce mediante un circuito oscilador (oscilaciones) que genera frecuencias superiores a las ultrasónicas, es decir, por encima de los 30 kHz. Estas oscilaciones se amplifican eléctricamente y se trasladan a una antena que será la encargada de liberarla a la nube.

Esta RF deberá ser modulada por circuitos electrónicos de forma tal que sea útil para transportar la información que interesa transmitir, sean datos digitales, tonos o voz. También será necesario un equipo receptor compuesto por una antena y un circuito demodulador para obtener la información de la señal de RF que es transmitida.

La potencia de la transmisión de RF será adecuada a la distancia que se encuentra el receptor, así como de las condiciones que influyan en la transmisión.

[5] Imagén obtenida en internet de la siguiente dirección:
http://www.medic.ula.ve/histologia/anexos/microscopweb/MONOWEB/capitulo2_1.htm

Se dice que hay línea de vista cuando no existen obstáculos entre transmisor y receptor en una ruta directa. Al no existir línea de vista, la transmisión es de tipo multi-ruta, la cual es característica de la comunicación por RF. En una transmisión de este tipo la señal sufre efectos como difracción, refracción, reflexión y dispersión, los cuales provocan que la comunicación entre transmisor y receptor se complete por diferentes trayectorias.

Las principales ***características*** de la comunicación por RF son:

- Facilidad con la cual puede ionizarse el aire para crear una trayectoria conductora a través del aire.
- Una fuerza electromagnética que conduce la corriente del RF a la superficie de conductores, conocida como efecto de piel.

El grado de efecto de estas características depende de la frecuencia de las señales.

Entre las ***ventajas*** que ofrece la comunicación por RF se tienen:

- Es una alternativa barata en aquellos lugares donde el cable no puede instalarse fácilmente.
- Es una opción para las comunicaciones portátiles.
- Por lo general no necesita ninguna licencia.
- Atraviesan paredes.
- Son omnidireccionales.
- Son capaces de transmitirse a grandes distancias.

CAPITULO VI: DISEÑO DEL HARDWARE Y SOFTWARE

6.1. IDENTIFICACIÓN DE PUNTOS A AUTOMATIZAR

Requerimientos:

Primeramente se requiere la automatización total de un proceso de reciclaje de plástico que consta de: tolva principal, faja de transporte, tolva secundaria y extrusor; mediante un control inalámbrico, en puntos específicos (críticos) del proceso, los cuales son los siguientes:

Tolva Principal: Se requiere la medición y el monitoreo del llenado de la tolva principal, con una distancia mínima de 5cm entre el sensor y el material de llenado para una tolva principal a escala de aproximadamente 30cm de alto en forma de embudo, además de una forma de controlar el ingreso del material a la tolva principal, para que en el caso de no haber material, esta se apague automáticamente o sea apagada.

Faja de Transporte: En este punto del proceso se requiere controlar la velocidad del motor de la faja haciéndola variable, para una faja de transporte a escala de aproximadamente unos 33cm de largo por unos 8cm de ancho, simulado por una pequeña faja de fotocopiadora, y su funcionamiento con un motor no mayor a 24Vdc.

Tolva Secundaria: Se requiere al igual que en la tolva principal la medición y monitoreo de la tolva secundaria, con una distancia mínima de 5cm entre el sensor y el material de llenado para la tolva secundaria a escala de aproximadamente unos 12cm. De alto en forma de embudo.

Extrusor: En este punto tal vez el más importante del proceso se requiere un motor no mayor a 24Vdc independiente al de la faja de transporte, que funcione cada vez que ingrese material al extrusor, se requiere también alguna forma para poder controlar la temperatura del calefactor del extrusor, en este caso simulado por una hornilla eléctrica de aproximadamente 660°c de máxima temperatura y de unos 1000w de potencia.

Para el control de mando:

Primeramente se requiere que sea una interfaz lo más dinámica posible y fácil de entender para un usuario final, se requiere una comunicación inalámbrica de un rango entre 10 a 50 metros lo que será más que suficiente para nuestro modelo a escala; además de encender y apagar el proceso desde el control de mando en el caso de una emergencia; dentro del menú del control de mando se requiere:

Menú Principal: Se requiere visualizar todos los parámetros monitoreables y variables del proceso, los que serán: tolva principal, tolva secundaria, ingreso de material, motor faja, motor extrusor, temperatura, además de una opción del consumo de batería. Dentro de cada opción del menú se requiere:

Menú Tolva Principal: Se requiere visualizar el llenado de la tolva en centímetros desde el menú principal, en un rango de 25 a 5cm de proximidad, para evitar el desbordamiento de material.

Menú Tolva Secundaria: Se requiere visualizar el llenado de la tolva en centímetros desde el menú principal, en un rango de 15 a 5cm de proximidad, para evitar el desbordamiento de material.

Ingreso de Material: Se requiere visualizar de alguna manera el ingreso de material a la tolva principal desde el menú principal, para que después de un tiempo de no ingreso de material, se apague el proceso para que no trabaje en vano.

Motor faja: Se requiere ingresar a la opción dada del menú principal y visualizar la velocidad del motor, para posteriormente poder variar dicha velocidad, dependiendo del requerimiento del proceso.

Motor extrusor: Se requiere ingresar a esta opción, desde el menú principal y visualizar un menú de activación de dicho motor, solo por el tiempo presionado o activado.

Temperatura: Se requiere ingresar a esta opción, desde el menú principal y visualizar la temperatura del calefactor, para posteriormente poder variarla dependiendo el requerimiento del proceso.

Consumo de batería: Se requiere visualizar el consumo del control de mando de la batería desde el menú principal.

El módulo en el que se automatizará el proceso es el siguiente:

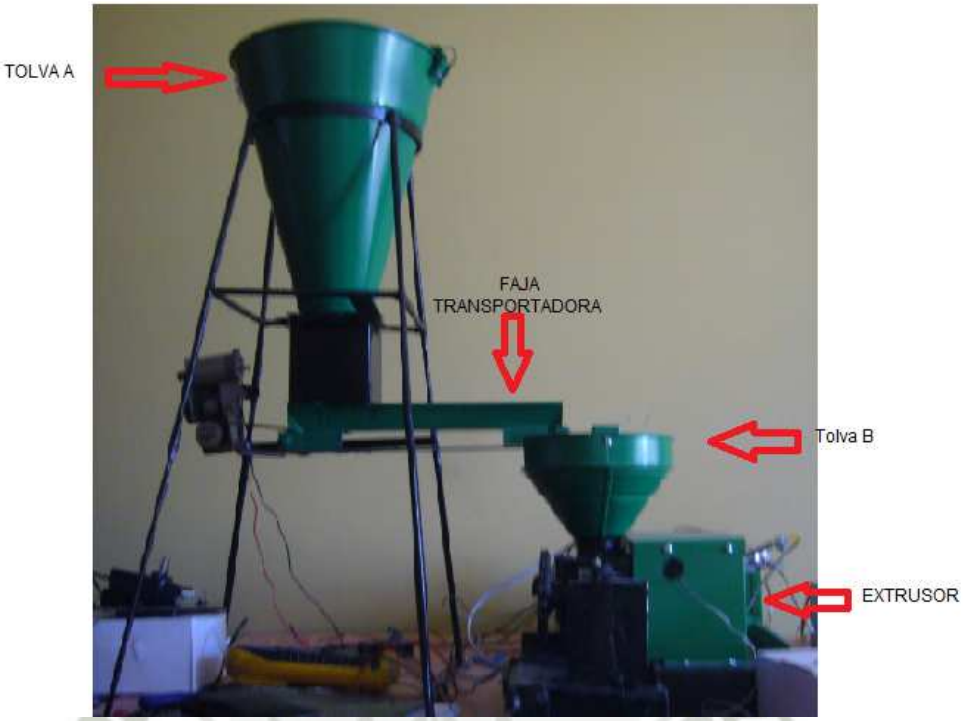


FIGURA 6.1. Modelo a escala del proceso a automatizar

6.2. FUNCIONAMIENTO GENERAL DEL PROCESO AUTOMATIZADO

El diseño genral del proceso automatizado por radiofrecuencia se muestra en dos diagramas de flujo; el primero corresponde a la Placa Maestra (Figura 6.2) y el segundo al Control de Mando (Figura 6.3)

FIGURA 6.2. Dia grama de flujo general de la Placa Maestra

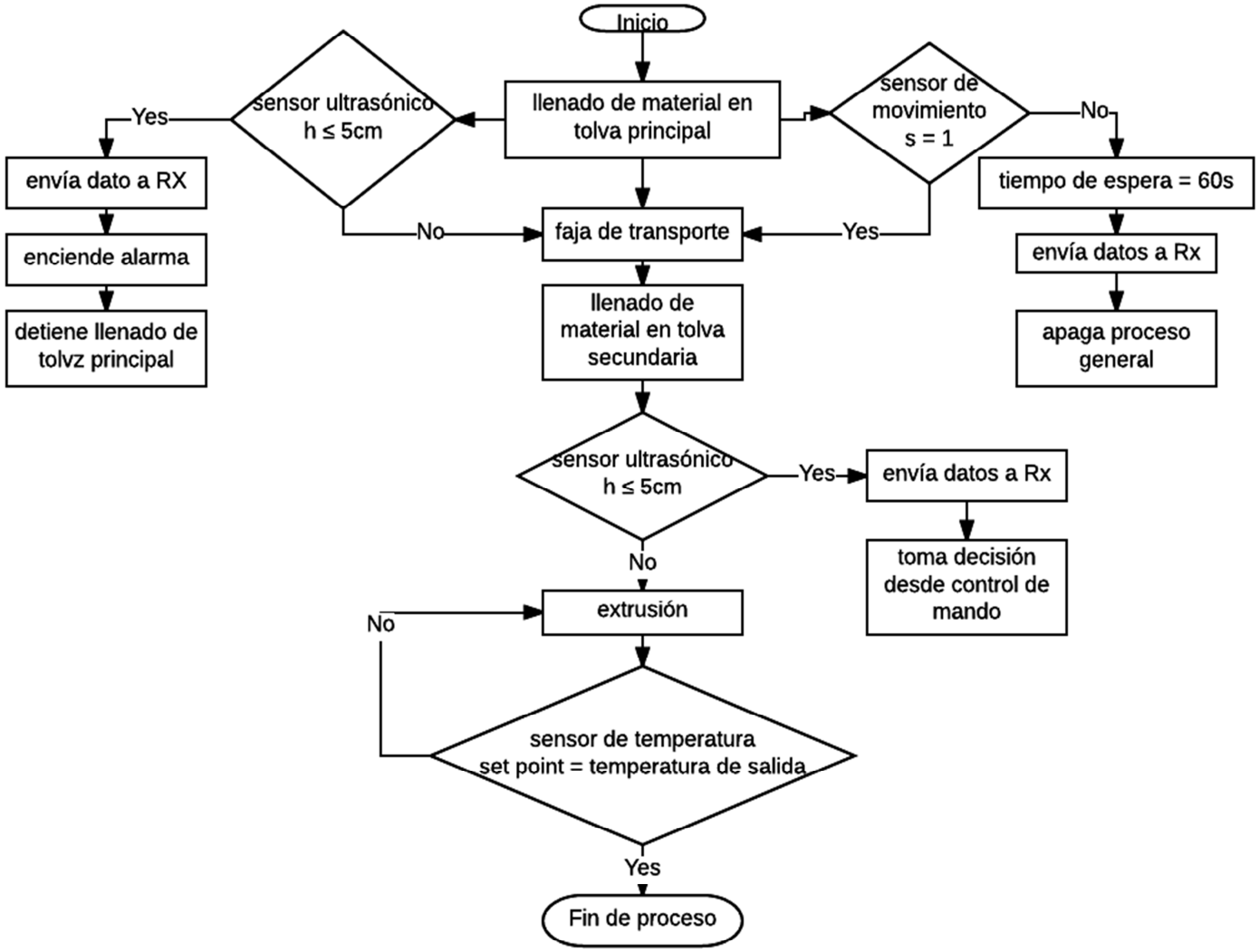
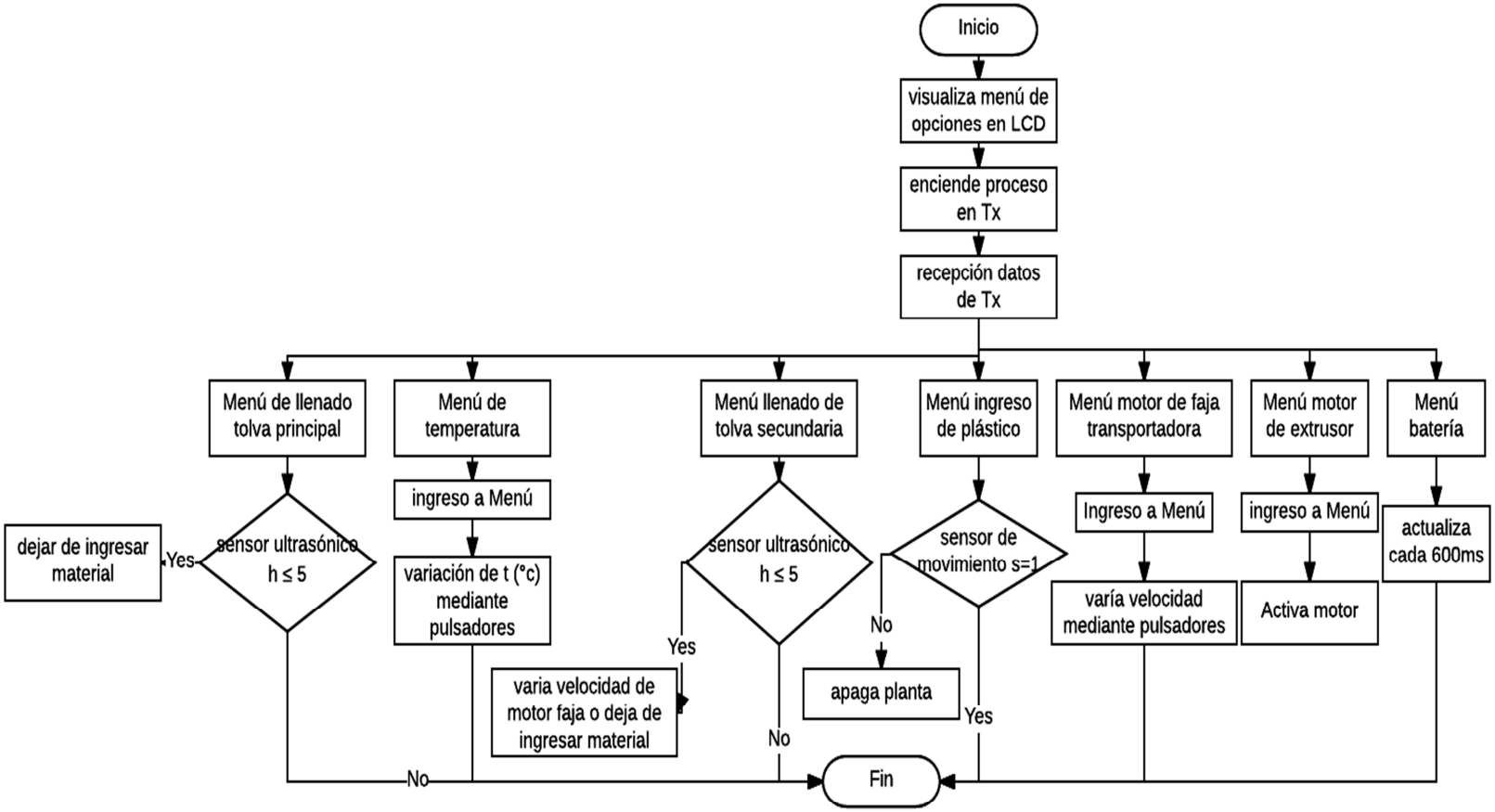


FIGURA 6.3. Dia grama de flujo general de la Placa Control de Mando



6.3. DISEÑO DEL PID

Para el control de la temperatura que se necesita en el calefactor, se está usando la tecnología PID:

PID (Proporcional, Integral, Derivativo)

El PID, es un mecanismo de control que a través de un lazo de retroalimentación permite regular, en este caso la temperatura de nuestro proceso. Lo que principalmente hace es calcular la diferencia entre la variable real, contra la variable deseada.

Existen varias formas de trabajar el PID mencionados anteriormente en el marco teórico, sin embargo para nuestro proceso se utiliza los tres parámetros: Ganancia P (proporcional), D (derivativa) e I (integral).

En este caso para lo que específicamente nos sirve cada uno de estos parámetros es: Proporcional (P), que sirve para tener una ganancia de la entrada en la salida; Derivativa (D), el parámetro derivativo, en este caso nos sirve para predecir el error de la temperatura que queremos obtener; finalmente el parámetro Integral (I), es la velocidad con que se va a dar la estabilidad de nuestro proceso; esta es una explicación muy básica de lo que hace el PID en nuestro proceso para querer variar la temperatura. Sin embargo el PID específico que se está usando en el siguiente proceso es el PID discreto.

PID DISCRETO

Para el proceso presentado, es necesario usar un PID discreto, ya que es necesario digitalizar la señal para así poder tener una automatización más exacta además que la variación que tiene el calefactor en temperatura no es constante, si no tiene una variación casi logarítmica, debido a que la temperatura tiene un tiempo muerto, después comienza a aumentar hasta otro cierto punto donde se estabiliza; esta representación de temperatura/tiempo la veremos más adelante al explicar el comportamiento mecánico del calefactor.

Es por eso que para poder automatizar y variar este punto del proceso (temperatura), es necesario que usemos el PID discreto, a continuación mostraremos el calefactor que se está usando en el módulo de prueba y como se halló su comportamiento .en el tiempo.

A continuación se muestra el calefactor usado para derretir el plástico por el método de extrusión:

FIGURA 6.4. Proceso de extrusión

Para enteder de una manera más sencilla el funcionamiento del PID se presenta a continuación un diagrama de bloques:

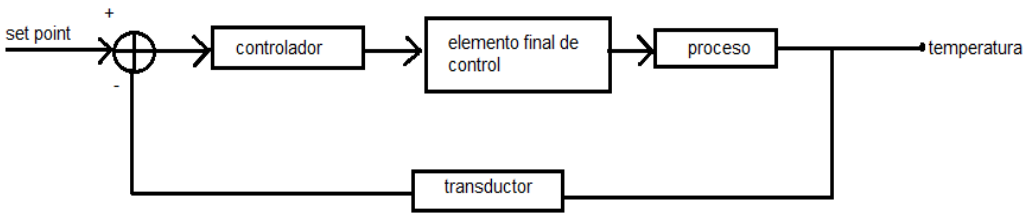


FIGURA 6.5. Diagrama de bloques general

Para poder variar la temperatura, es necesario saber el comportamiento del calefactor del extrusor, en otras palabras como varía la temperatura y hasta cuantos grados C° llega esta, y para poder saber ello necesitamos hacer pruebas las cuales se hicieron con el circuito de sensado, pero sin aplicar aún el PÍD, que se verá posteriormente; observando como varía la temperatura en una pantalla LCD y tomando el tiempo de acuerdo a una grabación hecha para esta medición.

Medición del calefactor en el extrusor para hallar el comportamiento del mismo:

FIGURA 6.6. Prueba para el comportamiento del calefactor

De acuerdo a los datos obtenidos en la medición del calefactor, tenemos un cuadro a continuación que compara la medición de la temperatura en relación al tiempo y que posteriormente con esos datos obtendremos una gráfica en donde se puede apreciar más claramente cómo se da el comportamiento del calefactor.

Los datos obtenidos se recopilaron en un video que grabó la pantalla LCD, y los datos obtenidos fueron subidos a Excel para tener un mejor orden y poder obtener la gráfica, aunque está también se hizo de forma manual para poder hallar las constantes que necesitamos para el PID.

A continuación podemos apreciar la tabla de la medición de la temperatura del calefactor en el tiempo:

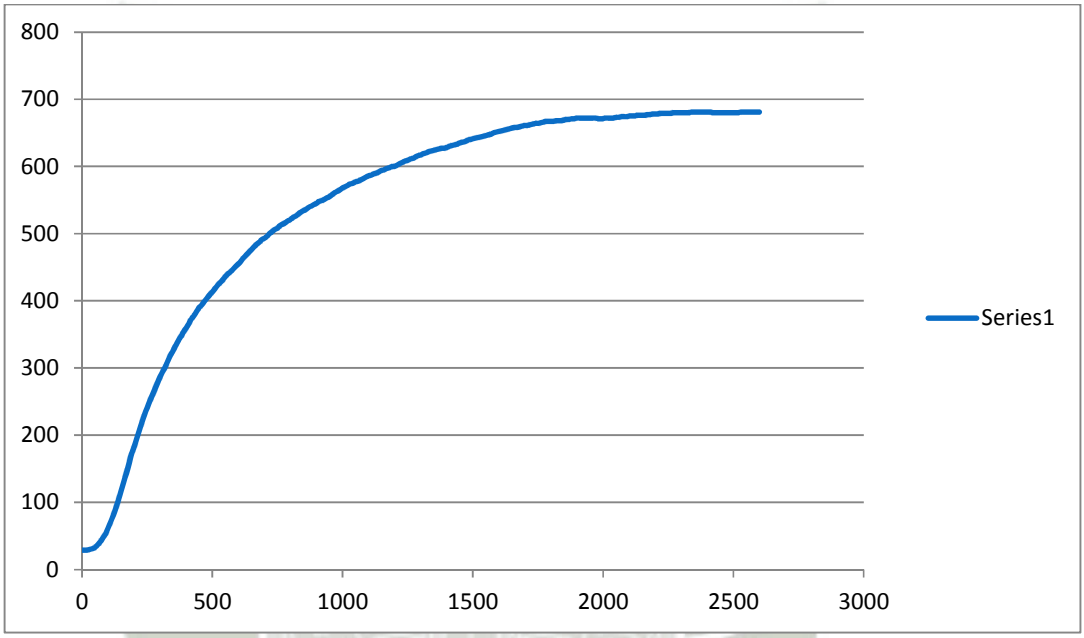
Tiempo (s)	Temperatura (°C)
0	29
100	62
200	183
300	287
400	360
500	413
600	455
700	493
800	521
900	545
1000	568
1100	586
1200	600
1300	617
1400	628
1500	641
1600	652
1700	661
1800	667
1900	672
2000	671
2100	675
2200	678
2300	680
2400	681
2500	680
2600	681

TABLA 6.1. Variación de temperatura en el tiempo del calefactor

Cabe resaltar que las mediciones que se tomaron fueron muchas más, exactamente 261 mediciones las cuales se puede apreciar en el anexo (anexo 1), esto se debe a que a más puntos de medición, es más exacto saber su forma de funcionamiento (cambio de temperatura), se

resumieron en 53 puntos de mediciones las que tenemos anteriormente para poder adjuntarlas en el informe de tesis y que no acapare espacio de más.

Ahora apreciaremos la gráfica de comportamiento que tiene el calefactor en el extrusor, y es así como varía su temperatura en el tiempo.



GRÁFICA 6.1. Comportamiento del calefactor en el extrusor

Como podemos apreciar en la gráfica anterior, el calefactor tiene un comportamiento logarítmico, lo más importante en ese comportamiento logarítmico es que el calefactor tiene un tiempo puede decirse muerto en el que no sufre cambios al inicio de la gráfica, luego en un punto comienza a variar ascendentemente la temperatura hasta que en otro punto casi a los 700 C° se estabiliza, este comportamiento se explicará de

forma mejor con la imagen de la gráfica hecha manualmente, donde también se hallan las constantes que el PID necesita.

GRÁFICA 6.2. Modelo para hallar las constantes del PID

Como podemos apreciar en la imagen anterior, se está hallando los puntos T1 y T2 en la gráfica logarítmica del comportamiento del calefactor para poder obtener las constantes.

Primeramente tenemos una gráfica en donde el plano X es el tiempo como se aprecia ahí y este está dado en segundos (s), y el plano Y es la temperatura dada en grados centígrados (C°), lo primero que se aprecia es un margen de temperatura muerto o sea en el que esta no varía donde

la temperatura es de 27 C°, casi hasta los 50 segundos, luego de este punto la temperatura varía un cambio no constante que llega casi hasta los 2300 segundos, donde la temperatura recién logra estabilizarse a unos 680 C° aproximadamente la cual es bastante alta para licuar el plástico es más a este punto lo quemaría es por eso que debe ser controlado; sin embargo para poder hallar las constantes del PID, es necesario usar un método, el cual es conocido como el método de Ziegler y Nichols, este método permite ajustar o "sintonizar" un regulador PID de forma empírica, sin necesidad de conocer las ecuaciones de la planta o sistema controlado y es el que estamos usando para poder hallar el comportamiento de nuestro calefactor.

Para calcular los parámetros se comienza por trazar una línea recta tangente a la señal de salida del sistema. Esta tangente está dibujada en la imagen con una recta, la cual se puede apreciar en el gráfico 6.6.

Además de estos dos tiempos característicos también hay que calcular la variación de la señal escalón dX y la variación de la respuesta del sistema dY.

El tiempo T1 señalado con la flecha inferior izquierda corresponde al **tiempo muerto**. Este es el tiempo que tarda el sistema en comenzar a responder. Este intervalo se mide desde que la señal escalón sube, hasta el punto de corte de la recta tangente con el valor inicial del sistema, que en este caso es el valor de 27 C°.

El tiempo T2 señalado en la parte superior de la imagen es el tiempo de subida. Este tiempo se calcula desde el punto en el que la recta tangente corta al valor inicial del sistema hasta el punto en el que la recta tangente llega al valor final del sistema.

Entonces T1 = 90s – 50s = 40s

T2 = 1000s – 90s = 910s

Además de estos dos tiempos característicos también hay que calcular la variación de la señal escalón dX y la variación de la respuesta del sistema dY.

En nuestro caso para el calefactor la variación de la señal escalón corresponde a dX = 100% de señal de control ya que estamos trabajando al 100% del comportamiento y la variación del sistema corresponde a dY = 680ºC – 27 °C medidos por el sensor, donde 680 es el punto máximo de temperatura al que llega el sistema, y 27 es el grado de temperatura en el que se inicia el comportamiento del calefactor.

A partir de estos valores se puede comenzar a calcular la constante del sistema Ko:

Dónde: Ko = (dx X T2) / (dy x T1)

Entonces: Ko = (100 X 910) / (653 X 40)

Ko = 91000 / 26120

Ko = 3.48

Con la constante Ko hallada se pueden calcular los parámetros del controlador PID con acción solo proporcional (P), proporcional e integral (PI) o proporcional integral y derivativa (PID).

Para el controlador P, PI, y PID se tiene que:

	K_P	T_i	T_d
P	Ko		
PI	0.9 x Ko	3.3 x T1	
PID	1.2 x Ko	2 x T1	0.5 x T1

TABLA 6.2. Constante Kp y Tiempos (Ti y Td) del PID

La constante Kp corresponde a la ganancia proporcional, Ti es la constante de tiempo integral y Td es la constante de tiempo derivativa. En el caso de tener el controlador PID configurado con las ganancias integrales Ki y derivativa Kd en vez de los tiempos Ti y Td, hay que tener en cuenta las siguientes relaciones entre ellos:

$$Ki = Kp / Ti$$

$$Kd = Kp \times Td$$

Con lo cual la tabla de valores para ajustar el controlador PID será la siguiente:

	K_P	k_i	k_d
P	Ko		
PI	0.9 x Ko	0.27 x Ko/T1	
PID	1.2 x Ko	0.60 x Ko/T1	0.60 x Ko x T1

TABLA 6.3. Constantes del PID

Donde:

dX = 100% - 0 = 100%

dY = 680 - 27 = 653 ºC

T1 y T2 fueron hallados anteriormente:

T1 = 40s

T2 = 910s

Entonces tenemos que para Kp en solo Proporcional (P):

Ko = 3.48

Para los valores Kp y Ki en Proporcional e Integral (PI):

Para Kp:

Kp = 0.9 (3.48) = 3.132

Para Ki:

Ki = 0.27 (3.48)/40 = 0.0235

Para los valores Kp, Ki y Kd en Proporcional, Integral y Derivativo (PID):

Para Kp:

Kp = 1.2 (3.48) = 4.176

Para Ki:

Ki = 0.60 (3.48)/40 = 0.0522

Para Kd:

Kd = 0.60 x 3.48 x 40 = 83.52

Seguidamente en la tabla siguiente se sintetizan los parámetros obtenidos del regulador PID.

	K_P	K_i	K_d
P	3.48		
PI	3.132	0.0235	
PID	4.176	0.0522	83.52

TABLA 6.4. Constantes halladas del PID

Con estos valores obtenidos podemos aplicar un PID al calefactor para poder automatizarlo y ya que matemáticamente se obtuvieron las constantes podemos estar seguros que la temperatura podrá ser variada con mayor exactitud; gracias a los valores constantes obtenidos para el PID.

Ahora para poder utilizar el PID discreto se tienen dos tipos de aproximaciones: rectángular y trapezoidal, de los cuales en este proyecto se optó por el segundo.

Aproximación Trapezoidal:

En este método el diseño se realiza en el dominio discreto directamente utilizando técnicas de ubicación de polos, a diferencia de la aproximación rectangular donde el diseño se realiza en el dominio analógico y a continuación se transfiere al dominio discreto; se optó por el método de aproximación trapezoidal, ya que este método es utilizado cuando se requiere una mayor precisión en la conversión discreta, y en este caso nuestro calefactor tiene un comportamiento inestable en el tiempo, es por

ello que se optó por este tipo de aproximación porque se necesita una mayor calidad y precisión en nuestro PID,

En la aproximación trapezoidal la integral se determina con la suma de trapezoides como se aprecia en la siguiente figura:

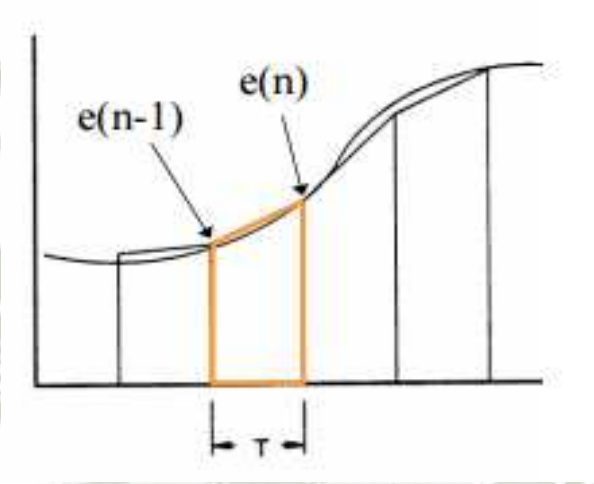


FIGURA 6.7. Método trapezoidal para el PID discreto[6]

En donde el área del trapezoide es:

$$A = \frac{T}{2}[e(n) + e(n-1)]$$

[6] Imagen obtenida de la siguiente dirección de internet:
http://es.slideshare.net/cesarcesitar/teoria-pid-control

6.4. DISEÑO DEL CIRCUITO DE CONTROL DE TEMPERATURA

Es importante resaltar, que en esta parte del informe por ser un modelo electrónico, muchas partes funcionan como un todo sin embargo se explicará individualmente los circuitos, y si se necesita explicar algo más específico o algo que explicar, pues este se encontrará al momento que hablemos del microcontrolador, ya que este componente es el ”cerebro” del circuito, y es el componente que hace los cálculos matemáticos, recibe datos los cambia, los arregla o por así decirlo, pero como ya se dijo el microcontrolador tiene un apartado donde sólo se habla de él.

Para poder apreciar y desarrollar el diseño del circuito se presenta a continuación el diagrama esquemático del circuito:

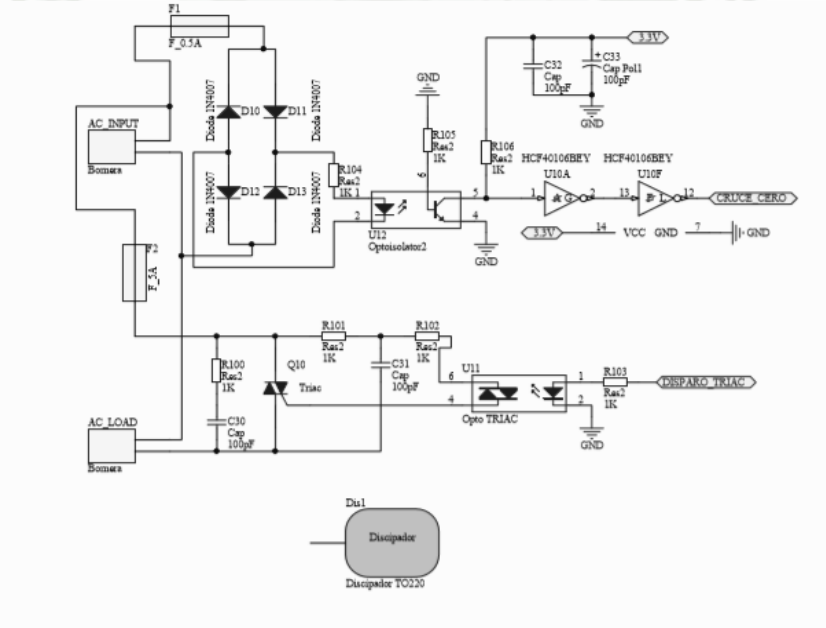


FIGURA 6.8. Esquemático del control de temperatura

Este circuito está diseñado para que la temperatura que varía en el calefactor pueda ser automatizada mediante el PID. Otro punto importante que se tiene que tomar en cuenta es la parte de la alimentación del calefactor, ya que este trabaja con corriente AC; para un parte se usa el circuito de detector de cruce por cero y por otra parte la del disparo del triac, además de tratar de aislar la parte de potencia de la parte electrónica, ya que muchos componentes son propensos a dañarse.

6.4.1. Detector de cruce por cero

A continuación mostraremos una parte del circuito de cruce por cero para desarrollarlo de manera más específica:

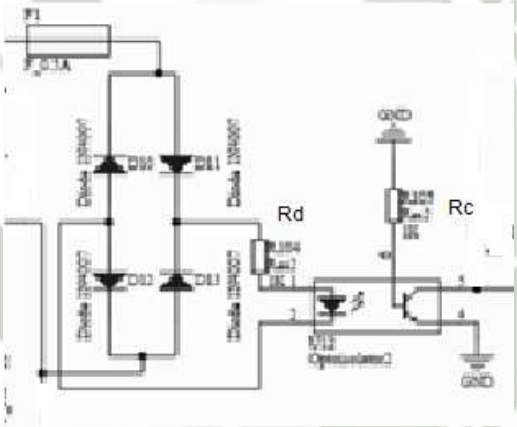


FIGURA 6.9. Esquemático del circuito de cruce por cero

Primeramente, se explica la función que cumple el puente rectificador de onda, que en este caso es un rectificador de onda completa, y lo que ocurre es lo siguiente:

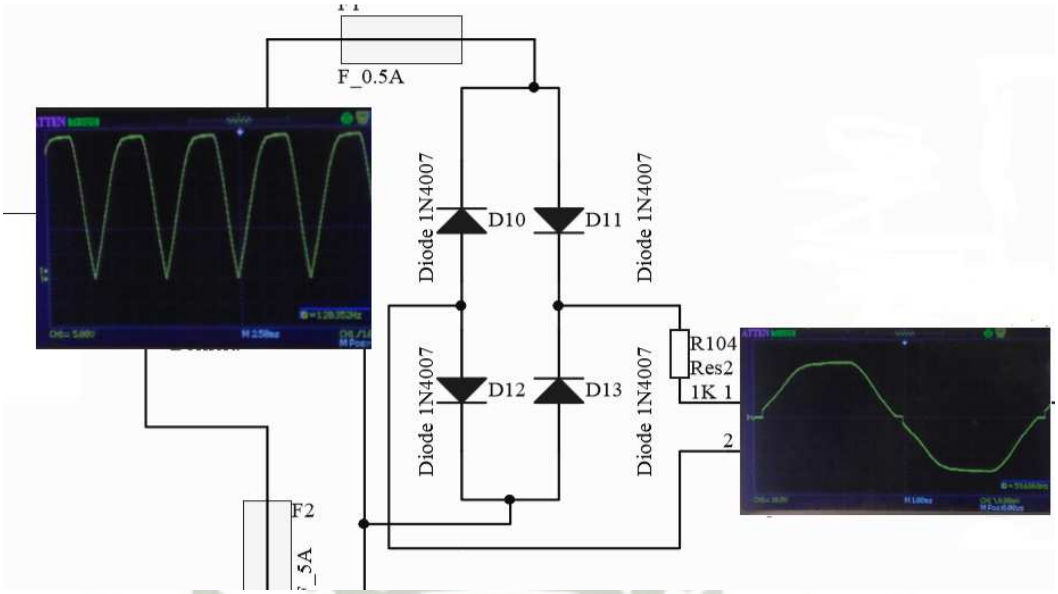


FIGURA 6.10. Señal antes y después del rectificador de onda.

Como se puede apreciar ingresa una onda senoidal de 220 Vac, y al pasar por el rectificador de onda completa esta se convierte en la onda rectificada con 311 Vpico en DC.

Ahora siguiendo con el circuito, lo que se hará a continuación serán cálculos matemáticos para hallar las resistencias, corrientes y en algunos casos potencias necesarias para el diseño del circuito.

Para comenzar con el diseño, lo primero que debemos hacer es revisar el datasheet; según las referencias en el datasheet del optoacoplador 4n35, los valores máximos a los que puede trabajar el optoacoplador y los que refiere el datasheet pueden causar daños permanentes al dispositivo, ya que estos son solo rangos de estrés, y esto no implica la operación funcional del dispositivo en estas condiciones. En otras

palabras la exposición a condiciones de máximos-absolutos durante periodos prolongados puede afectar a la fiabilidad del dispositivo.

Entonces teniendo en cuenta esto para hallar Rd en el circuito antes mostrado, se tiene que el valor máximo de corriente en el diodo es Id = 60mA, así que probaremos con una corriente media Id = 30mA.

Para el cálculo de Rd en el circuito de cruce por cero:

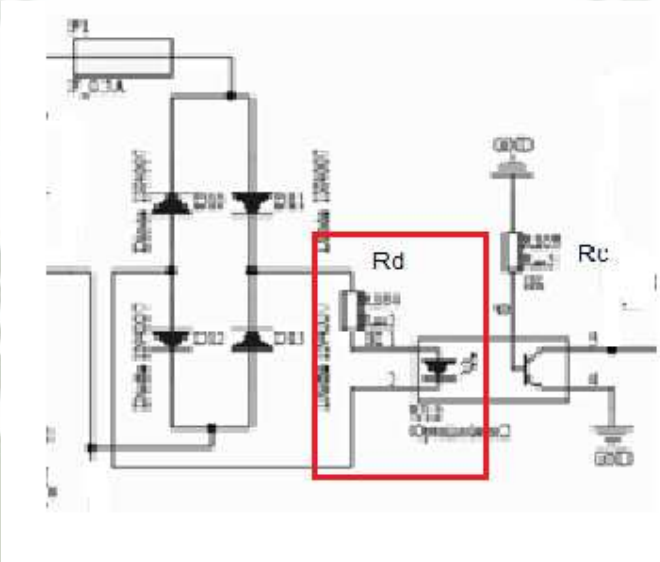


FIGURA 6.11. Rd en el circuito del cruce por cero

Según el datasheet y lo que se explicó anteriormente para el diodo mostrado dentro del cuadro rojo, la corriente en él es 30mA entonces en Rd se tiene que:

$$Vd = Rd \times Id$$

Donde se tiene que al ser rectificada la onda completa, a la salida del rectificador se tiene que:

$$Vmáx = Vs \times \sqrt{2}$$

Dónde:

$$Vs = 220V$$

Entonces tenemos que:

$$Vmáx = 220 \times \sqrt{2}$$

$$Vmáx \approx 311\, V = Vd$$

Con Vd = 311V e Id = 30mA (según restricciones de datasheet anteriormente mencionado) tenemos en Rd que:

$$Vd = Rd \times Id$$

$$311 = Rd \times 30mA$$

$$Rd = 10.37K$$

Hallamos la potencia en Rd para saber si nuestra resistencia elegida es la correcta:

$$Pd = Vd \times Id$$

$$Pd = 311 \times 30mA$$

$$Pd = 9.33\ W$$

Como podemos apreciar la potencia en Rd es muy alta, y para no tener problemas en este punto del circuito mejor trabajar a una potencia menor, es por ello que la resistencia que antes hallamos era 10.37K, y la que utilizamos en el circuito es una mayor a ese valor en este caso Rd = 22K, para que la corriente que pase por Rd sea mucho menor y por ende la potencia también.

Entonces hallamos Id con Rd = 22K:

$$Vd = Id \times Rd$$

$$311 = Id \times 22K$$

$$Id = 14.14mA$$

Entonces para Pd:

$$Pd = Vd \times Id$$

$$Pd = 311 \times 14.14mA$$

$$Pd = 4.4W$$

Entonces se trabaja prácticamente con una corriente de 15mA y con una Rd grande y potente de 10W, para que pueda trabajar en una zona de reposo sin problemas.

También hallaremos el tiempo en el que el cruce por cero está siendo detectado en un voltaje instantáneo en este caso de Vinst = 11V.

$$11 = 22K \times Id$$

$$Id = 0.5mA$$

Entonces para detectar el tiempo del cruce por cero se sabe que en una gráfica senoidal:

$$Vinst = Vmax \operatorname{sen}(\omega t)$$

Sabiendo que:

Vinst = Tensión instantánea

Vmax =Voltaje máximo

ω = Velocidad angular

t = Tiempo

Valor Máximo (Vmax)

Sabemos que w = 2 π f y la frecuencia de la corriente alterna en el Perú es de 60Hz:

$$w = 2\pi (60)$$

$$W = 377 \text{ rad/s}$$

Entonces en la fórmula general:

$$Vinst = Vmax \text{ sen } (\omega t)$$

Reemplazamos:

$$11 = 311 \text{ sen } (377 \times t)$$

$$0.035 = \text{sen } (377t)$$

$$t = 5.32 \text{ ms}$$

Siguiendo con el diseño de nuestro circuito; en el optoacoplador 4n35; en la parte del transistor, la que se muestra a continuación:

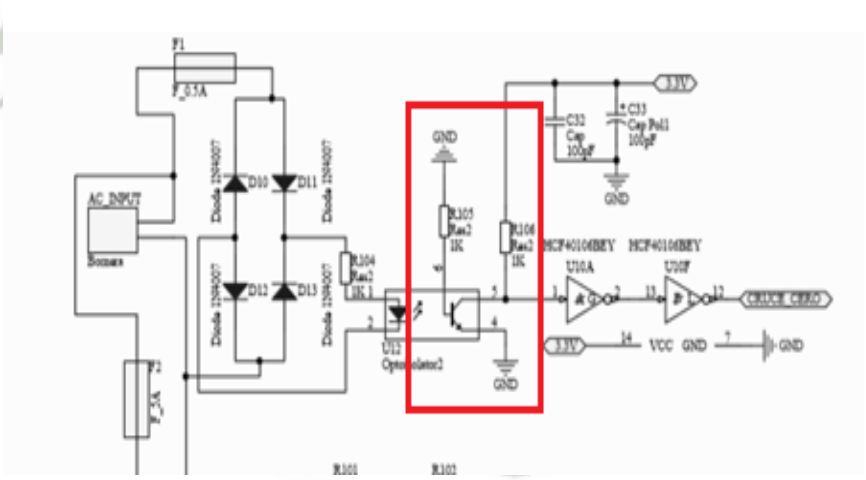


FIGURA 6.12. Optoacoplador en el circuito de cruce por cero

Para calcular Rc en el transistor, se sabe gracias al datasheet que según la relación en la curva de entrada en el diodo Id y la corriente Ic en el colector:

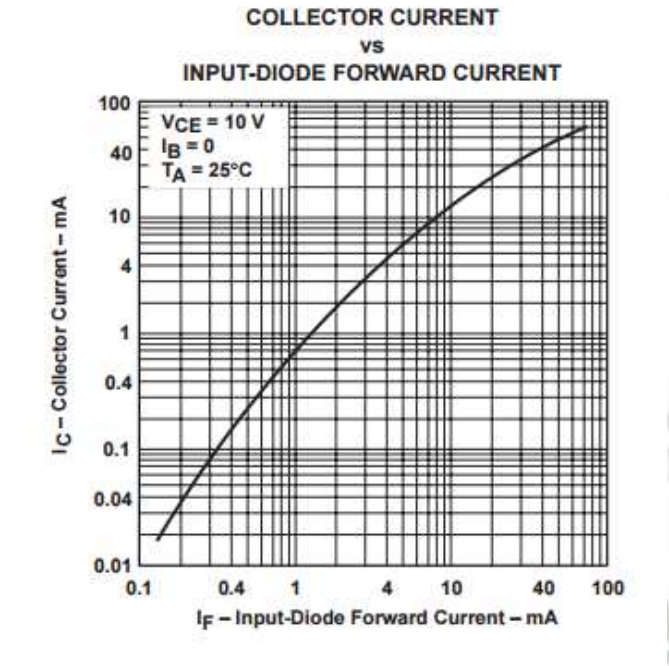


GRÁFICO 6.3. Relación entre Id e If para diseño[7]

Como se puede apreciar en la gráfica, para una corriente Id = 15mA que es la que estamos utilizando, la aproximación en Ic ≈ 20mA.

Esto quiere decir que cuando Id=15mA, Ic= 20mA y donde Id ≈ Ib.

Entonces en Rc, tenemos que Vc=3.3V e Ic=20mA

$$Vc = Ic \times Rc$$

$$3.3 = 20mA \times Rc$$

$$Rc = 165\ \Omega$$

[7] Gráfica entre la corriente If e Id obtenida del datasheet del componente optoacoplador 4n35.

Para que exista una corriente mucho menor en Ic e Id, la resistencia que se está utilizando en el circuito es Rc = 11.8K

Entonces:

$$3.3 = Ic \times 11.8k$$

$$Ic = 0.28\ mA$$

Según la gráfica en el datasheet entre Ic e Id, hallamos que Ib ≈ Id = 0.5mA

Para Rd:

$$311 = Rd \times 0.5mA$$

$$Rd = 622\ K$$

Seguidamente tenemos las señales antes y después del optotriac:

Señal después del puente rectificador antes del optotriac

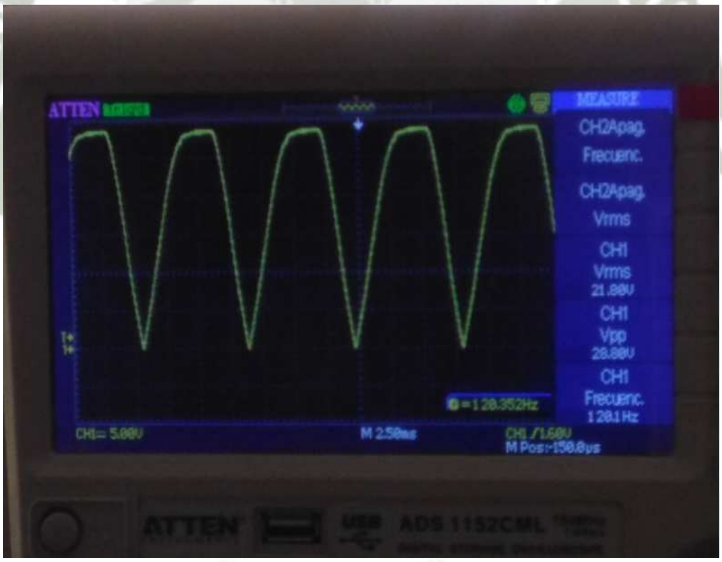


FIGURA 6.13. Señal rectificada después del puente de diodos

Después de pasar por el optotriac, la señal que sale en colector es la siguiente:

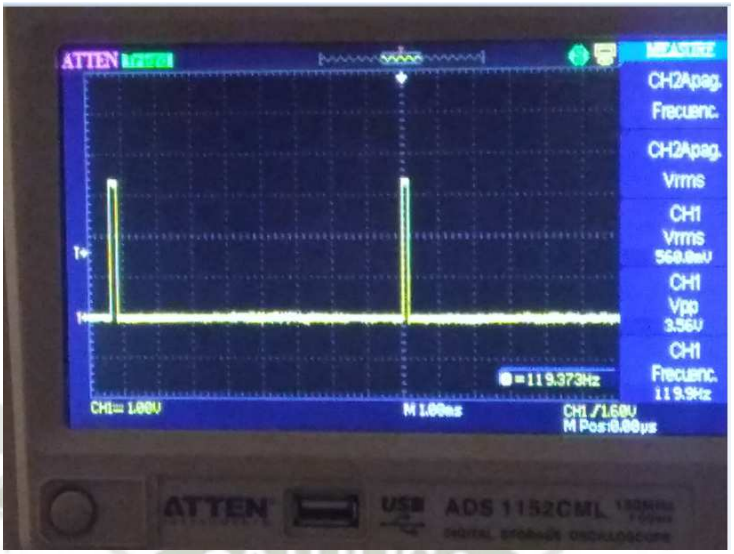


FIGURA 6.14. Señal del cruce por cero para poder ser reconocida

De esta manera en cada pulso u onda cuadrada, se detecta el cruce por cero.

Finalmente para una carga de 5A, se está eligiendo un triac de 16A, con ello es más que suficiente para la carga y para que este no caliente y trabaje en una zona de reposo.

Para Id= 15mA (corriente óptima ya que su corriente máxima es de 60mA) en el optotriac de la carga:

$$3.3 = 15mA \times Rb$$

$$Rb = 220\ \Omega$$

6.4.2. Control de potencia

Para la otra parte del diseño del cruce por cero, el que podemos apreciar en la siguiente imagen:

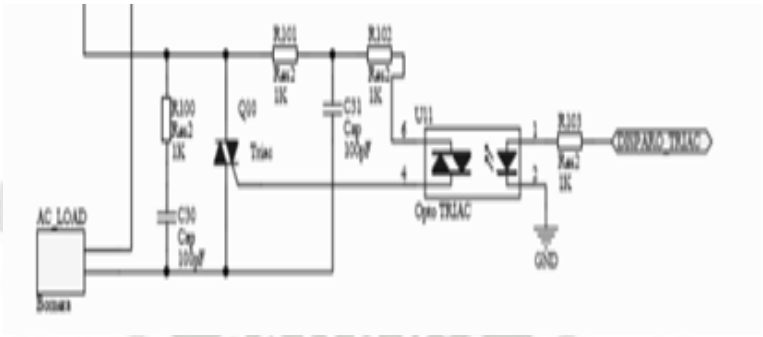


FIGURA 6.15. Esquemático del control del disparo del triac

Para la parte del optotriac MOC 3021, se tiene que revisar el datasheet, entonces para el diseño de este circuito se está usando los datos mencionados por el fabricante, en este caso como encontramos el datasheet una red snubber para el optotriac y otra red snubber solamente para el triac , esto se hace para que no disparen en falso sobre todo cuando hay cargas inductivas.

Para poder saber los valores del circuito snubber, veremos a continuación el esquemático obtenido del datasheet:

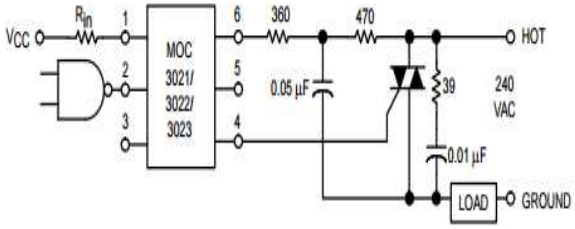


FIGURA 6.16. Esquemático obtenido del dathaseet del moc3021[8]

[8] Diagrama esquemático obtenido del datasheet del componente optotriac moc3021

Las redes snubber sirven para protección de los triacs; se utilizan para suprimir transitorios indeseables y eliminar problemas en los circuitos de conmutación con elementos **inductivos** y capacitivos. La conmutación en estos circuitos puede producir EMI (interferencias electromagnéticas) que afecten a otros equipos y si no se suprimen las sobretensiones transitorias se pueden exceder los límites de los dispositivos y producir su degradación o destrucción. Con la utilización de redes snubbers en los circuitos con tiristores se pretende, por un lado amortiguar los posibles picos transitorios de tensión y por otro que la derivada de tensión en los bornes del tiristor quede por debajo de un determinado valor límite.

Este optoisolador no debe utilizarse para conducir una carga directamente, está destinado a ser un dispositivo de disparo único. En este circuito el lado "caliente" de la línea está conmutada y la carga conectada a frío o al lado de tierra.

La resistencia de 39 ohmios y el capacitor de 0,01 µF son para amortiguar el triac, la resistencia de 470 ohmios y el capacitor de 0,05 µF son para ignorar el acoplador. Estos componentes pueden o no ser necesarios dependiendo de la particular del triac y de la carga usada.

6.5. DISEÑO DEL CIRCUITO PARA EL CONTROL DE MOTORES

Para poder controlar los motores, se está utilizando PWM (modulación por ancho de pulso). La forma más eficiente de variar la velocidad de un motor es modificando la frecuencia; es por eso que se crearon los equipos

basados en este principio que son los variadores de velocidad que son una de las aplicaciones más comunes de puentes inversores. Estos equipos toman la onda de corriente alterna suministrada por la red eléctrica a una frecuencia fija de 50 o 60 Hz y las cambian a otras frecuencias según la necesidad de la aplicación, normalmente se logran rangos de 5 a 400 Hz.

Este variador de velocidad es muy beneficioso para nuestros motores ya que nos permite un amplio rango de condiciones de operación y además reduce la energía requerida para su arranque permitiendo que este se haga en forma gradual hasta alcanzar la velocidad y potencia plenas.

Los motores a controlar son los siguientes:

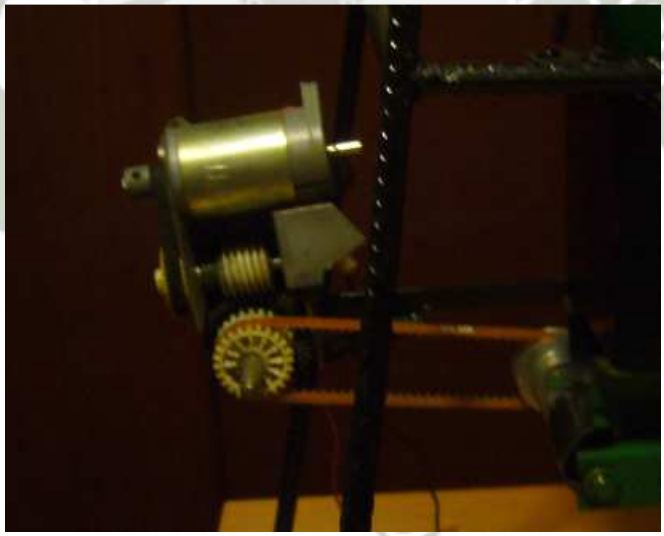


FIGURA 6.17. Motor para el control de faja transportadora

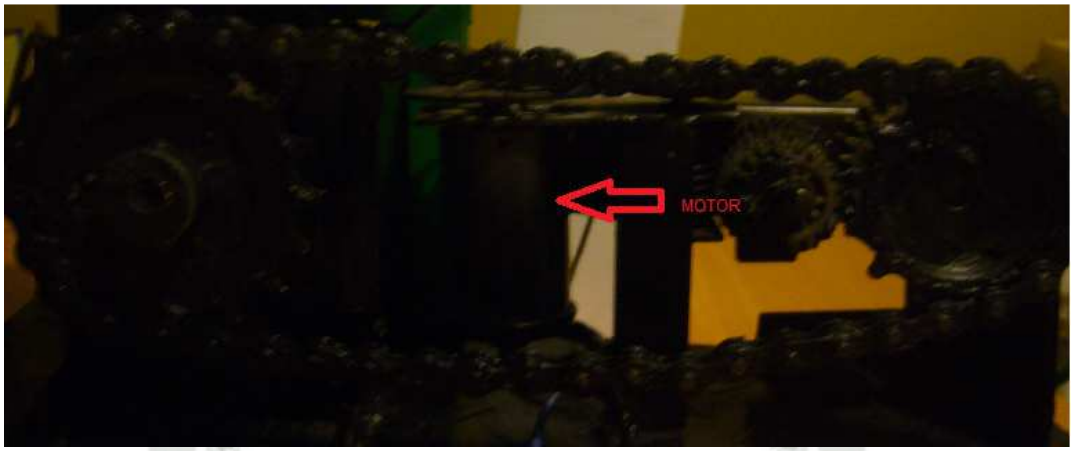


FIGURA 6.18. Motor para el control de extrusión

Para el diseño del circuito es el mismo que se encuentra en el datasheet como “typical connection”, sin embargo el cambio que se le hizo en este proyecto es que el ir 2110 no lo estamos usando como fuente flotante, si no que esos dos puntos q salen del mosfet están simplemente yendo a tierra; para este diseño que ya existe en el datasheet se calcularon las dos resistencias que van de la salida del integrado al mosfet, se puede apreciar mejor en la siguiente imagen:

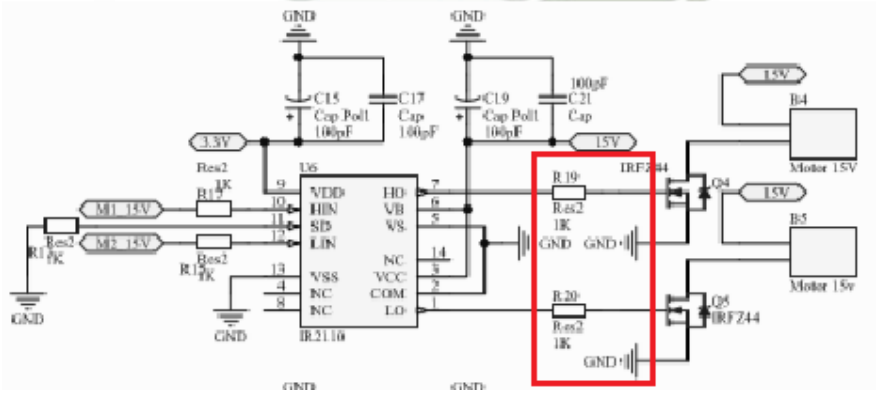


FIGURA 6.19. Esquemático del circuito del control de motores para el diseño de Rm

Para el cálculo de estas resistencias se tiene la ecuación general de la ley de ohm donde:

$$V = I \times R$$

Como se ve en el datasheet a la salida del ir2110 se tiene una corriente Io = 2 A, entonces:

$$15 = 2\ A \times Rm$$

$$Rm = 7.5\ \Omega$$

Para el cual tenemos el valor comercial de 10 Ω

Para Rin a la entrada del ir2110, como podemos ver en la siguiente imagen:

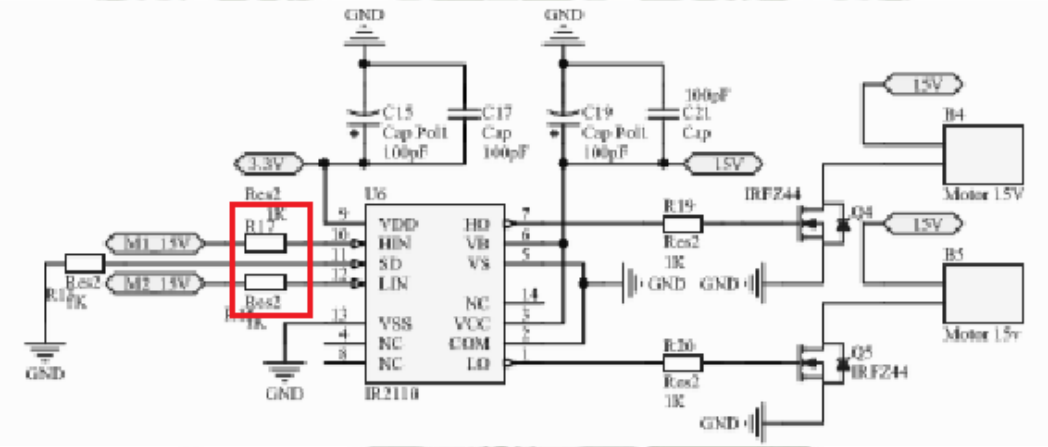


FIGURA 6.20. Esquemático del circuito del control de motores para el diseño de Rin

Dónde: i_L= 40uA, Vin = 3.3 V

Se sabe que i_L= 40uA, gracias al datasheet donde la corriente de polarización de entrada lógica “1” máxima es de 40uA.

Entonces, se tiene que:

$$3.3 = 40u \times Rin$$

$$Rin = 82.5k$$

La corriente de 40uA al ser despreciable podemos aumentar ese valor a un valor que se adapte más al diseño y a los valores del mercado; entonces para diseño, el ir 2110 al pedir poca corriente, el integrado tiene alta impedancia, entonces elegimos un valor adecuado para Rin como 1K.

$$3.3 = i_L \text{ x } 1k$$

$$i_L = 3.3mA$$

Al recibir esos 3.3mA no existen problemas.

6.6. MÓDULOS INALÁMBRICOS

El pin (OUT del XBee) va conectado al pin (Rx del microcontrolador) y el pin (IN del XBee) va conectado al pin (Tx del microcontrolador) **la conexión de este pin no se debe realizar directamente entre el microcontrolador y el XBee por lo que el microcontrolador entrega 5V y el XBee trabaja a entre (2.8 a 3.3)V,** por lo tanto se debe realizar un ajuste de voltaje por medio de un regulador o divisor de voltaje que adecúa la señal.

FIGURA 6.21. Módulo XBEE en el transmisor

FIGURA 6.22. Módulo XBEE en el receptor

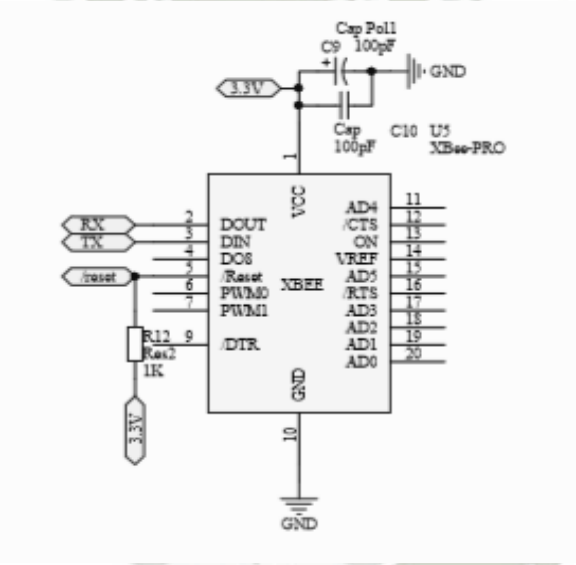


FIGURA 6.23. Esquemático del módulo XBEE

Al trabajar con estos módulos RF, lo que hacen primeramente es crear una red zigbee, esta red zigbee está compuesta por 3 tipos de nodos o modos de trabajo, los cuales se explican a continuación:

6.6.1. Nodos de trabajo

1. **Coordinador**

   Solo puede existir un coordinador, al trabajar en este modo el xbee en modo coordinador cumple las siguientes funciones:

   - Encargado de formar la red de trabajo que, en este caso, es una red PAN. Si no hay un coordinador no puede existir nada, así que obligatoriamente un xbee tiene que estar en modo coordinador.
   - Encargado de manejar las direcciones de todos los nodos en la red.
   - Maneja los aspectos de seguridad en la red.
   - Maneja también la auto regeneración de la red, así como otras funciones más.
   - Puede funcionar como un intermedio entre dos dispositivos (como un repetidor), indica el canal que se va a usar.
   - No se le permite dormir (modo sleep), o sea que debe mantenerse siempre prendido.

2. **Router**

   Puede existir uno o más ruteadores, y tiene las siguientes funciones:

   - Se une a una red previamente ya formada por algún coordinador.
   - Puede mandar y recibir sus propios datos, provenientes de sus propios puertos.

- “Rutea”, esto quiere decir que sirve de intermediario entre dos nodos distantes, o sea enlaza dispositivos finales (end device) con el coordinador; en otras palabras, sirve para administrar la red.
- Su uso es opcional si no hay nodos distantes que estén fuera de alcance uno del otro.

3. Dispositivo Terminal (end device)

Puede existir uno o más dispositivos terminales, cumple las siguientes funciones:

- Es una versión simplificada de un ruteador.
- Al igual que el ruteador, el dispositivo terminal se una a una red previamente formada por el coordinador.
- Manda/recibe sus propios datos, es decir sus datos provenientes de sus puertos locales.
- Los dispositivos terminales necesitan de un coordinador o ruteador para unirse a la red y tener la capacidad de mandar y recibir datos, es decir que en este modo de trabajo del xbee (dispositivo terminal), siempre necesita de un intermediario para poder interactuar con la red zigbee.
- Además, usa menos hardware que los ruteadores o los coordinadores y también consume menos potencia.
- También pueden trabajar de manera intermitente, o sea estos dispositivos terminales permanecen la mayor parte del tiempo en

estado inactivo o dormidos y despiertan intermitentemente por periodos muy cortos de tiempo para enviar o recibir datos y luego nuevamente regresan a su estado de descanso o inactividad (sleep), es por esto que los dispositivos terminales consumen menos potencia.

6.6.2. Configuración de módulos XBEE

Para poder configurar nuestros módulos XBee es necesario contar con el software X-CTU, seguidamente se debe instalar el driver FT232 por administrador de dispositivos (solamente cuando se conecta el módulo XBee al PC por el puerto USB, realizar esto con anterioridad).

Al ingresar al software, se elige el COM# correcto, posteriormente pasa a la ventana de Terminal y digita la letra (B), luego se regresa a la ventana PC Setting verifica la elección del COM# y el Baud por defecto 9600, pero si ha tenido una configuración anterior, se debe leer al Baud con que fue configurado 9600 u otro valor de Baud Rate dentro del intervalo que trae el X-CTU. (ver figura).

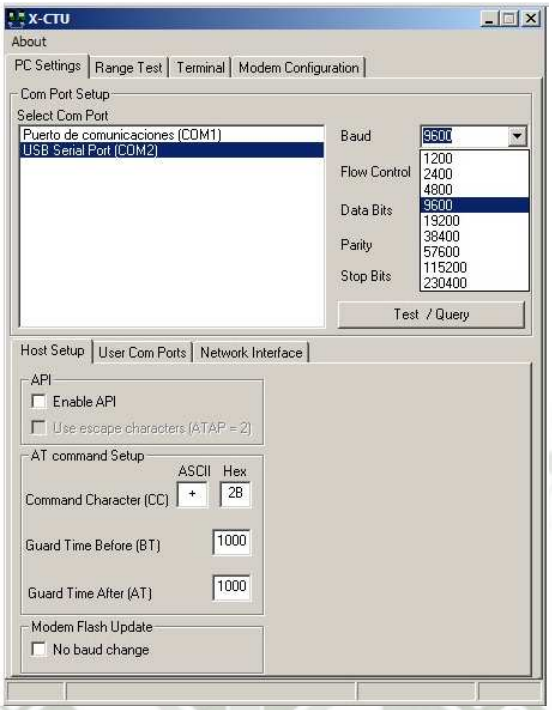


FIGURA 6.24. Interfaz de configuración inicial para módulos XBEE del progama X-CTU

Siempre se debe verificar el Baud; porque el problema radica que al tomar un módulo XBee de fábrica por defecto el Baud es de 9600, pero si no es así el Baud se pudo haber modificado con anterioridad y es por ello que se debe intentar con las nueve opciones entre (1200 a 230400) Baud, teniendo en cuenta lo mencionado anteriormente, este problema ocurrió con uno de los módulos XBee, y se tuvo que cambiar y probar con todas las opciones.

Seguidamente, ir a Modem Configuration y dar click en Read, como podemos apreciar en la figura; luego a Restore (así se restauran los parámetros de fábrica), ir a la ventana PC Setting y leerlo nuevamente (no requiere salir del X-CTU). Se configuran los siguientes comandos:

- En la parte de (Function Set) se selecciona ZIGBEE ROUTER AT. Como vemos a continuación en la siguiente figura.

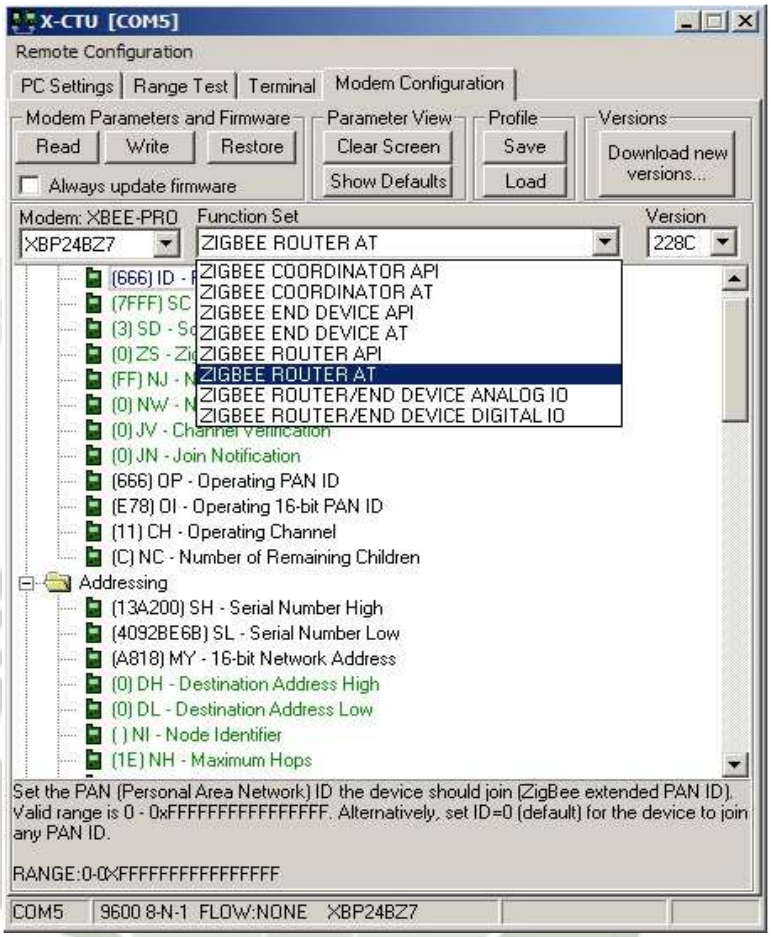


FIGURA 6.25. Configuración de XBEE en modo router

- ID-PAN-ID (Es la red de trabajo, se coloca un número de 3 ó 4 cifras, este número deberá ser igual en ambos módulos XBee). Como observamos a continuación.

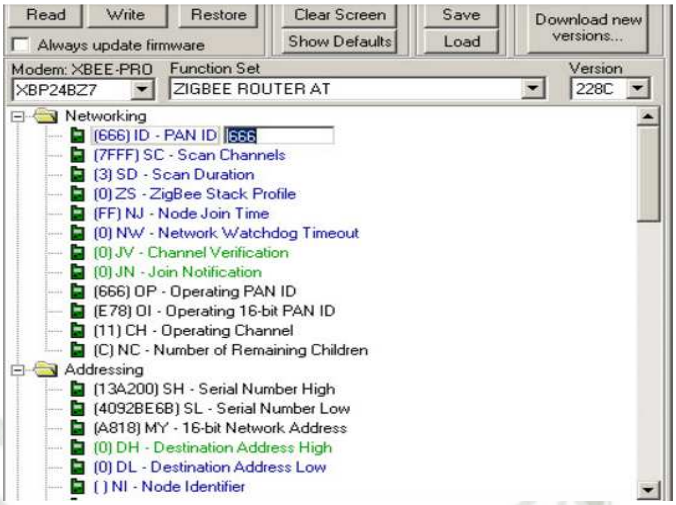


FIGURA 6.26. Configuración de parámetros en el XBEE

- DH (Dirección de destino alta, se coloca 13A200 se refiere al número único de los módulos Xbee igual para ambos). En este caso podemos encontrar también la alternativa de dejarlo como (0) y este se refiere a que se comunicará al Coordinador.
- DL (Dirección de destino baja, se coloca el número único de serie del módulo Xbee al cual va a transmitir, este número lo encuentra en la parte de debajo de cada módulo). En este caso se puede encontrar también la alternativa de dejarlo como (0) y este se refiere a que se comunicará al Coordinador.
- BD (Baud Rate “es la velocidad en que la comunicación cambia de estado en un periodo”), se selecciona la opción 3 a 9600 igual para los dos módulos, como podemos apreciar en la siguiente figura,

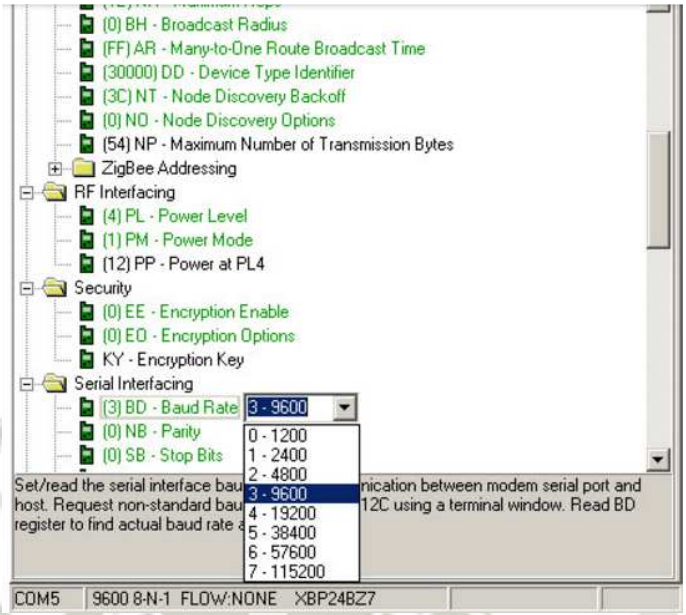


FIGURA 6.27. Configuración de baudrate en el XBEE

Se debe tener en cuenta también los siguientes parámetros (No modificarlos), ya que para el caso de no tener éxito con la comunicación estos serían un punto de partida importante a tener en cuenta.

(11) CH - Operating Channel, para este caso el número 15, varía dependiendo el canal que tome automáticamente el módulo XBee, indica el canal de operación aunque no es un parámetro ajustable se tiene en cuenta para que sea el mismo tanto en el Router como en el Coordinador. En la opción mostrada a continuación:

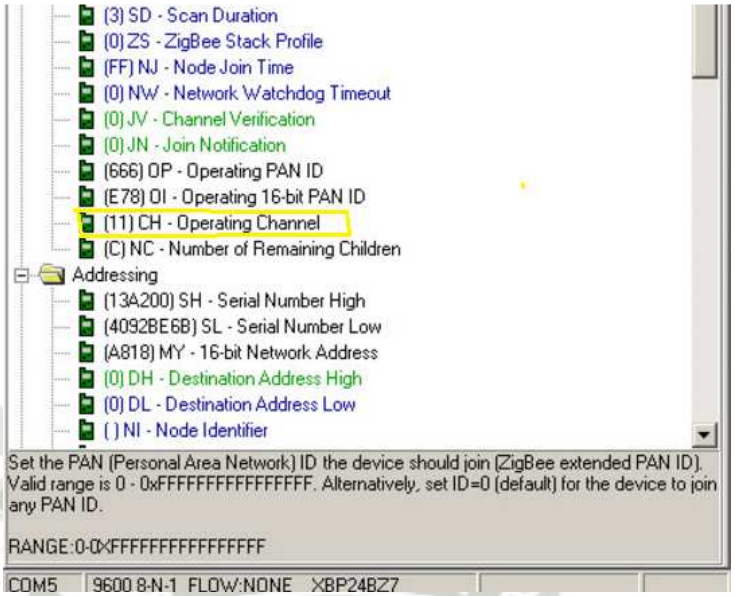


FIGURA 6.28. Configuración del canal de operación en el XBEE

Luego, se le da escribir (Write), y aparecerá en la parte de abajo un OK.

Tener en cuenta que esta configuración se realiza igual para los dos módulos, sino le aparece OK, volver a restaurar los parámetros de configuración desde el punto de modem configuration**.**

Para seguir con la configuración, en la ventana (Terminal), se da click en Hide Hex, esta parte sirve para visualizar los datos que se envían y reciben del módulo XBee, en la parte izquierda se visualiza un código Ascii y en la derecha código Hexadecimal.

Una opción importante es enviar paquete de datos dando click en Assemble Packet (en este puede escribir en los dos códigos anteriores un paquete de datos y luego enviarlos).

Los datos (de dos caracteres), que aparecen en la parte derecha de la interface, corresponden al código HEX (es genérico para cualquier PC) con 256 combinaciones (8 bit), como se muestra en la siguiente tabla.

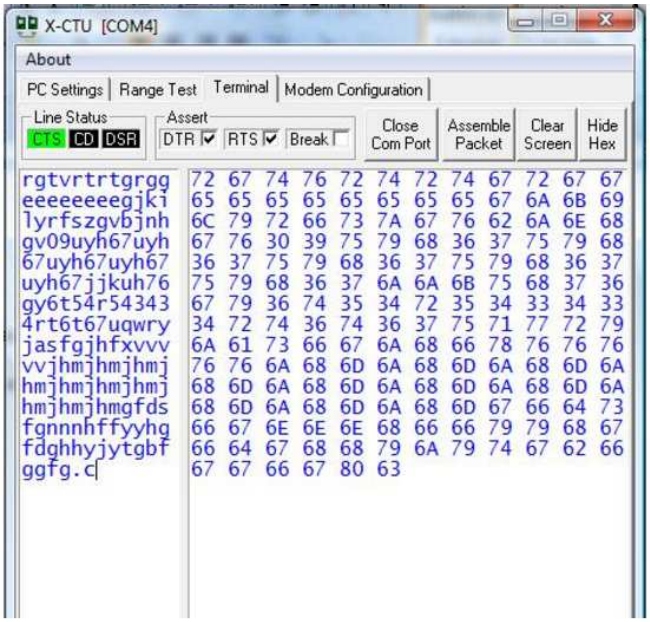


FIGURA 6.29. Combinaciones de 8 bits en código HEX en el módulo XBEE.

Una vez que la configuración y el envío de datos estén funcionando correctamente (esto solo se puede verificar si ya está configurado el módulo XBee Coordinador del otro equipo), se procede a hacer las pruebas respectivas para la verificación.

La manera más efectiva de hacer pruebas es completar la siguiente Tabla 1. Para este paso a cada grupo se le asignará un código diferente para que realice la comunicación, es decir, que se le asignara un dato especificado aleatoriamente en código Hexadecimal, Ascii ó

binario (8 bit), en el cual, deberán transmitir por medio de la interface X-CTU y verificarlo en los 8 led de la recepción. (Tabla 2).

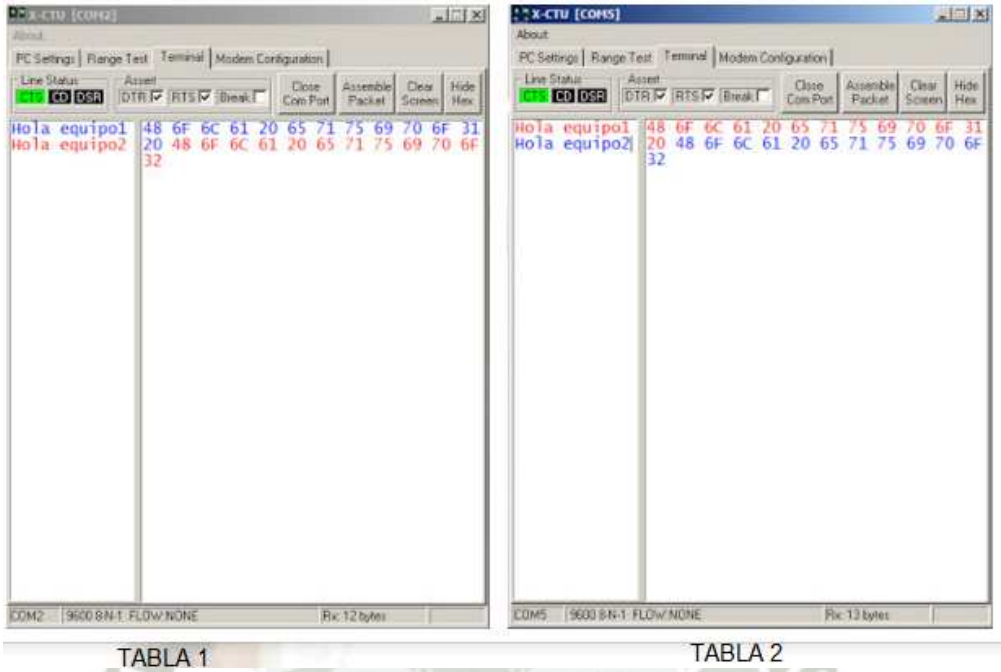


FIGURA 6.30. Pruebas de envío de datos entre módulos XBEE

Nota: Una vez configurado y revisado cada uno de los parámetros y pasos de configuración se procede con una previa verificación en la ventana de Terminal, se da click en Hide Hex y en cada equipo con PC separados se comunican entre sí Coordinador y Router. Para la anterior de las tablas se optó por tomar el pantallazo de las dos interface X-CTU que representan el Coordinador y Router en una sola PC. Se debe tener en cuenta que quien transmite se distingue por la letra azul y quien recibe el mensaje por la letra roja.

Por otro lado, en el caso de nuestro proyecto estamos usando simplemente un Coordinador el cual creará la red PAN, y un dispositivo

terminal, ya que no tenemos mayor necesidad de contar con un ruteador porque nuestra comunicación es directa simplemente de transmisor a receptor, en los dos módulos RF. Cabe resaltar que sí se puede dar la opción de que en algún momento se necesite un ruteador y tal vez más de un dispositivo terminal, pero eso ya dependería de las condiciones y variaciones ya sea del lugar, ambiente o la forma de automatización misma de cómo, cuándo y dónde se quiera hacer.

Otro punto del modo de funcionamiento xbee que debemos tener en cuenta, es según el intercambio de datos con el módulo. En el caso de nuestros datos a enviar estamos usando el llamado modo transparente, porque lo que necesitamos en esta planta a escala es un módulo de prueba y lo que se quiere es enviar y recibir datos simples y con eficacia, para lo que es más que suficiente usar el intercambio de datos en modo transparente.

Ya que el puerto puede transmitir datos provenientes de entrada de puerto serial, análoga o digital, el módulo xbee principal (coordinador), enviará los datos recopilados por cada sensor, como el del sensor PIR de movimiento, los ultrasónicos para saber el llenado de las tolvas, y el de temperatura; todos estos datos son enviados al dispositivo terminal donde llegan sin pérdidas y los visualizamos claramente en la pantalla LCD que se muestra a continuación:

FIGURA 6.31. Pantalla LCD en el control de mando para visualización de parámetros

Finalmente, al visualizar lo resultados sensados, podemos variar mediante el control inalámbrico los parámetros necesarios para una producción con mayor eficacia y eficiencia, que los métodos ya antes conocidos en automatización.

Observaciones: Los módulos XBee Serie 1 y Serie 2B tienen el mismo pin-out. Sin embargo, los módulos Serie 1 no pueden comunicarse con los módulos Serie 2B. Si está utilizando esto fuera de Estados Unidos, para eso se debe revisar las leyes con respecto a las radiocomunicaciones dependiendo del país.

6.7. FUENTES DE ALIMENTACIÓN Y REGULADORES DE VOLTAJE

Este punto del diseño, se refiere además de las fuentes que se usan las cuales son dos fuentes variables, también tenemos los reguladores de voltaje ya que no todos los dispositivos electrónicos trabajan a un mismo voltaje, es por eso el uso de ellos.

a) Fuente de alimentación 0-15V.

Esta fuente se usa para hacer funcionar a todos los componentes electrónicos, como el microcontrolador, los integrados, en general hacer funcionar los circuitos.

FIGURA 6.32. Fuente variable de 0 a 15v.

b) Fuente de alimentación 0-24V.

Esta fuente se usa específicamente para trabajar solo con los motores, ya que antes se tuvo una sola fuente para todo en general y se tuvo un problema de trasientes.

Este problema de los trasientes, trata de que los armónicos de los motores dañaban la placa, estos trasientes eran generados en la red de alimentación, para arreglar este problema se optó por aislar las fuentes, además de cambiar el regulador que trabaja a 3.3V por un LDO de mayor velocidad.

FIGURA 6.33. Fuente variable de 0 a 24V.

6.8. DISEÑO DE REGULADORES DE VOLTAJE

En este proyecto estamos usando varios reguladores de voltaje, debido a que no todos usan el mismo voltaje para su funcionamiento, ahora apreciaremos el circuito que estamos usando para el regulador de voltaje:

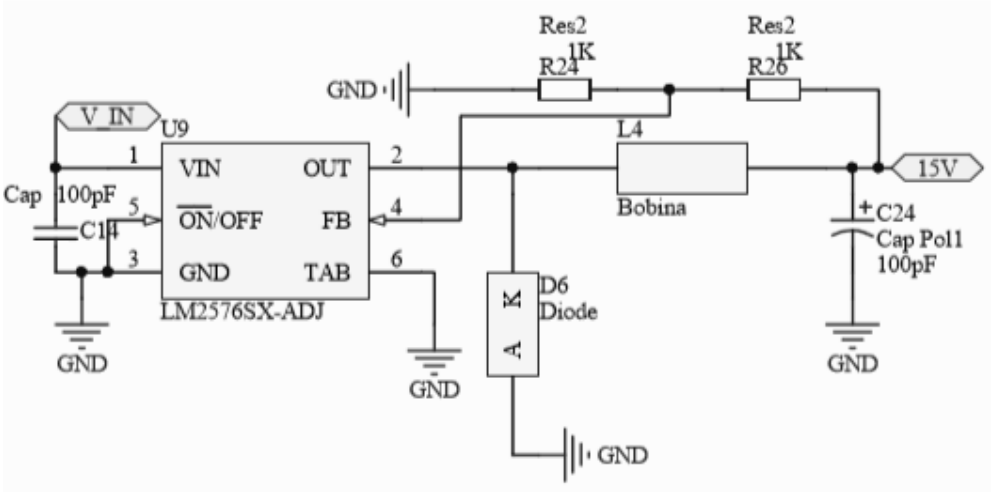


FIGURA 6.34. Esquemático del regulador de voltaje de 15V.

Para lo que es diseño, se tiene que resaltar que en el datasheet del LM2576, se encuentra el diseño del circuito regulador de voltaje tanto para 3.3, 5, 12 y 15 voltios. Aun así se tiene que hacer los cálculos respectivos, los que se encuentran a continuación:

En esta configuración de voltaje regulable, para el cálculo de resistencias se tiene que:

$$Vout = Vref\ (1 + R2/R1)$$

Entonces tenemos que:

$$R2 = R1\ (Vout/Vref - 1)$$

Donde, gracias al datasheet sabemos que:

$$Vref = 1.23V.\ y\ R1\ está\ entre\ 1K\ y\ 5K$$

Para nuestro diseño tomaremos que

$$R1 = 2.8K,\ 2.7K\ valor\ comercial$$

Además también sabemos que el voltaje de salida requerido es de 10V.

Entonces tomando en cuenta que R1 = 2.8K

$$R2 = (2.8K)(10V/1.23V - 1)$$

$$R2 = (2.8K)(8.13V - 1)$$

$$R2 = (2.8K)(7.13)$$

$$R2 = 19.96K$$

Para el cual el valor comercial será R2 = 20K

Seguidamente se tiene que para el regulador de 5v se tiene en el datasheet el diseño del circuito que será usado, cabe resaltar que el mismo regulador de 3.3v se usa en el control de mando del proyecto, se usa solo el de 3.3V ya que en el mando no existe otro componente que trabaje a un voltaje diferente, el esquemático del circuito podemos verlo a continuación:

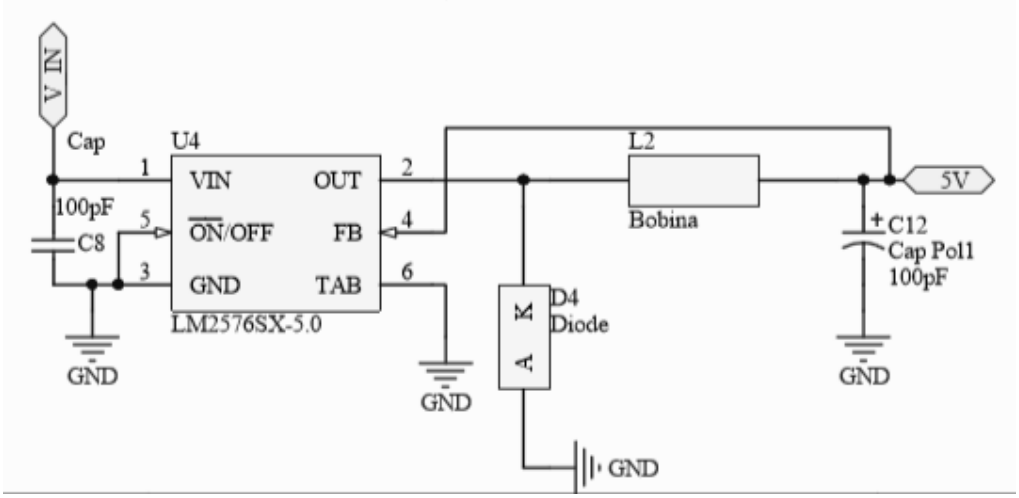


FIGURA 6.35. Esquemático del regulador de voltaje de 5V.

Regulador en el mando de 3.3 voltios

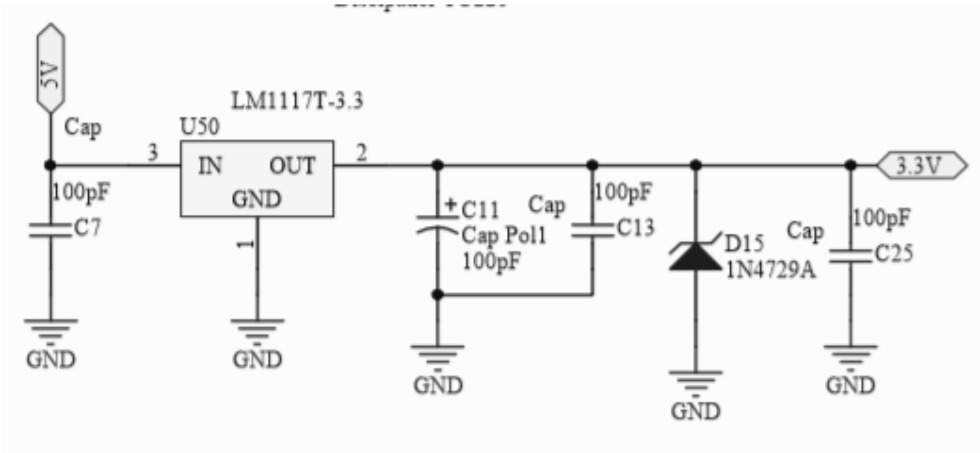


FIGURA 6.36. Esquemático del regulador de voltaje de 3.3V en el control de mando

Seguidamente se podrá apreciar el circuito que se encuentra en el datasheet el cual ya está diseñado así por él.

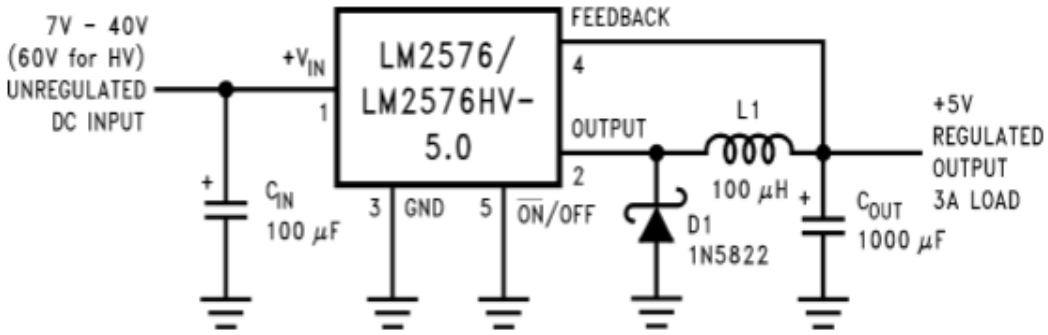


Figure 1.

FIGURA 6.37. Esquemático del diseño del regulador de voltaje en el datasheet del LM2576[9]

Finalmente el último regulador de voltaje es con el que anteriormente se tuvo problemas al momento de hacer las pruebas finales; el problema que

[9] Esquemático obtenido del datasheet del componente LM2576.

se tuvo es el de trasientes, los trasientes quieren decir que es el armónico de los motores los que dañaban la placa, estos se generan en la red de alimentación; lo que hizo que varios de los componentes y muchos de ellos importantes en la placa se dañen, hasta dos veces. Para poder resolver este problema se separaron las fuentes una para la parte electónica y otra para trabajar con la parte de motores, además de haber cambiado el regulador que trabaja a 3.3V por un LDO.

FIGURA 6.38. Trasientes generadas por los motores

Este último regulador LDO, trabaja a 3.3v. en su salida, esto quiere decir que trabaja con componentes bastantes sensibles a un pico de voltaje, que por el hecho de trabajar con 3.3V, es difícil que soporten siquiera 5V.

Todo se puede encontrar en la página 11 del datasheet, donde se encuentra prueba de circuito y diseño de dieléctricos.

6.9. SENSORES ULTRASÓNICOS Y DE MOVIMIENTO

6.9.1. Sensores Ultrasónicos:

Los sensores que estamos usando en este proyecto son del código HC-SR04; este módulo de medición ultrasónico proporciona una función de medición de 2cm – 400cm sin contacto, con una precisión de hasta 3mm. Los cuales podemos apreciar a continuación:

FIGURA 6.39. Sensor ultrasónico HC-SR04

Estos módulos incluyen transmisor, receptor y control de circuito.

El principio básico del control es el siguiente:

- El uso del trigger (disparo) IO da al menos 10us de señal de nivel alto.
- El módulo envía automáticamente ocho 40KHz y detecta si existe un pulso de la señal de vuelta
- Si la señal de retorno, en nivel alto, es el tiempo de duración de salida alta, es el tiempo de envío del ultrasonido al regreso.

Test de distancia = tiempo de nivel alto X velocidad del sonido (340m/s /2).

El módulo no puede estar a menos de 5cm. de distancia, ya que en este punto ya no tiene la calidad con la que trabaja a más de 5cm.

Estos sensores se están utilizando para poder monitorear el llenado de las tolvas tanto como la A y B, como se ve a continuación en la siguiente imagen:

FIGURA 6.40. Sensores ultrasónicos en Tolvas principal y secundaria respectivamente

Se optó por este tipo de sensores primeramente por cumplir con los requerimientos necesarios para el monitoreo de las tolvas, además de su óptimo funcionamiento y por tener un precio razonable en el mercado.

6.9.2. Sensor de movimiento

Este sensor PIR es un dispositivo piro eléctrico que mide cambios en los niveles de radiación infrarroja emitida por los objetos a su alrededor a una distancia máxima de 6m. El sensor se muestra a continuación en la siguiente imagen:

FIGURA 6.41. Sensor PIR de movimiento

Como respuesta al movimiento, el sensor cambia el nivel lógico de un "pin", por lo cual su uso es bastante simple; además es un sensor de bajo costo y de tamaño pequeño muy utilizado en sistemas de seguridad como alarmas, iluminación controlada por movimiento, también en aplicaciones de robótica y en este caso en automatización de una planta.

Otras de sus características en su salida el estado del pin es TTL, la polaridad de activación de salida es seleccionable, tiene un mínimo tiempo de calibración.

Este sensor cuenta con 3 terminales, dos de ellos para la alimentación y el último es la salida de detección de movimiento; para la conexión al microcontrolador solo requiere el uso de su única salida del sensor, a continuación podemos apreciar la conexión del sensor PIR:

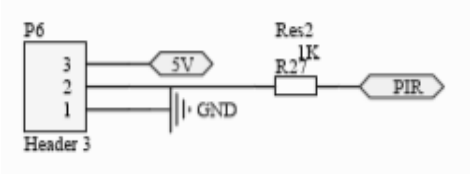


FIGURA 6.42. Esquemático del sensor de movimiento PIR[10]

[10] Esquemático obtenido del datasheet del sensor PIR de movimiento

Al ser energizado el sensor requiere de un tiempo de preparación para comenzar a operar de forma adecuada, esto se da debido a que el sensor tiene que adaptarse a las condiciones del ambiente, en este tiempo el sensor aprende a reconocer el estado de reposo o no movimiento del ambiente, y está calibración puede durar entre 10 a 60 segundos, y es altamente recomendable la usencia de personas en su foco mientras este se calibra.

6.10. EL MICROCONTROLADOR ATMEGA1284P Y SOFTWARE DE AUTOMATIZACIÓN

6.10.1. Control de ángulo de disparo

Para poder hacer el control del ángulo de disparo, anteriormente se vieron los circuitos que son de cruce por cero y de disparo de triac, también anteriormente se hizo una medición del calefactor para así poder saber el comportamiento del mismo, entonces al tener el comportamiento del calefactor, podemos automatizar la temperatura mediante cálculos matemáticos dentro del microcontrolador, el cruce por cero y el disparo del triac.

Al conocer el comportamiento del calefactor y saber las equivalencias tanto en tiempo como en ángulo, lo primero con lo que trabajamos para el ángulo de disparo es el ya conocido triac , este es un dispositivo capaz de conducir en ambas direcciones es por ello que es útil para usarlo en control de potencia de corriente alterna cual es el caso ahora;

Entonces al variar el ángulo de disparo, podemos variar la potencia que se suministra a la carga, esta parte lo hace el circuito ya mostrado antes que se ve en la siguiente imagen:

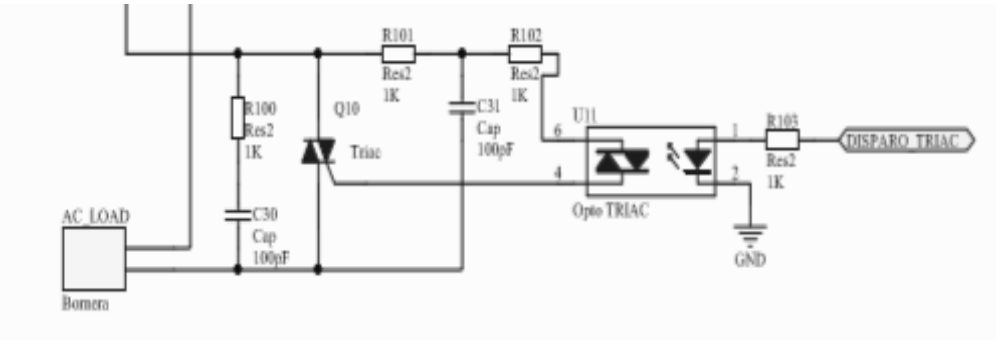


FIGURA 6.43. Control de disparo del triac

Ahora esa salida del microcontrolador llega a disparar el triac y lo que sucede es lo siguiente:

- Detecta inicio de un nuevo medio ciclo de onda.
- Esperar un tiempo t correspondiente al ángulo.
- Enviar la señal de disparo al circuito de control.
- Volver a detectar el inicio de un nuevo medio ciclo de onda.

Para poder detectar el inicio de un medio ciclo de onda usaremos el circuito siguiente cuya salida estará conectada al pin de interrupción externa:

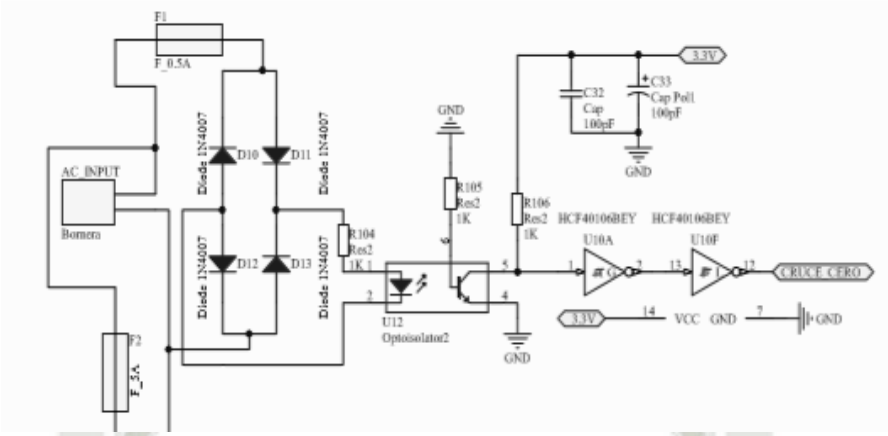


FIGURA 6.44. Cruce por cero

Este circuito se encarga de acondicionar la señal de la red para que el microcontrolador detecte cada medio ciclo de la red, por medio de la interrupción externa. A continuación podemos ver la conexión de los circuitos de potencia al pic:

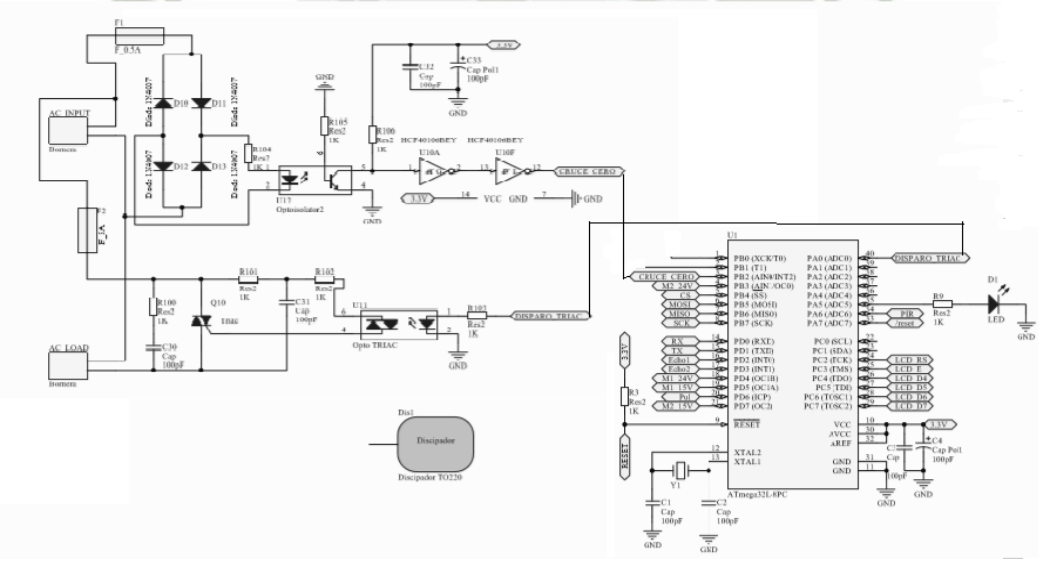


FIGURA 6.45. Esquemático del control de temperatura con el microcontrolador

El punto principal para este circuito detector del cruce por cero, es detectar el incio de cada medio ciclo de onda, para poder sincronizar nuestra señal con el disparo del triac.

6.10.2. PWM

Como se vio anteriormente el circuito con el que se está trabajando para los motores, como sabemos PWM es una onda rectangular de periodo fijo y como la frecuencia es la inversa del periodo entonces también será de frecuencia fija, y lo que varía es el tiempo en alto a lo que se llama ancho de pulso.

En el microcontrolador, el temporizador timer2 es de 8 bits, a la cantidad de bits que se utilizan se le llama resolución, entonces se dice que el módulo PWM AVR tiene una resolución de 8 bits, esta será nuestra resolución de la señal PWM.

Si la resolución es de 8 bits entonces el máximo será 255 y si la resolución es de 16 bits su máximo será 65535 el registro TCNT2 aumentará sus valores de 0 a 255. El aumento de sus valores en una unidad del registro TCNT2 puede ser con cada ciclo de trabajo del microcontrolador AVR, o si se utilizan los prescaler del timer2 el aumento en una unidad de sus valores tardará mas, dependiendo del prescaler utilizado.

En el PWM timer2 AVR modo rápido el registro TCNT2 irá de 0 hasta el máximo elegido, luego se reiniciará a 0 para volver hasta su máximo y luego otra vez de 0 a su máximo y así continuará en el

modo rápido; a este ir de 0 a su máximo le tomará un tiempo que dependerá del prescaler utilizado para el timer2, y ese tiempo que le tome al registro TCNT2 para ir de 0 a su máximo será el periodo de la señal PWM timer2 AVR.Una cosa a tener en cuenta es que al pasar el registro TCNT2 de máximo a 0 es como si se desbordará, por lo cual se puede habilitar el uso de la interrupción por desborde del timer2.

Normalmente el registro TCNT2 irá aumentando su conteo con cada ciclo de reloj del microcontrolador, si se usa por ejemplo una frecuencia de trabajo (lo que se conoce como FCPU) de 1Mhz entonces el registro TCNT2 aumentará en una unidad cada microsegundo, y dependiendo de la resolución utilizada, este registro aumentará desde un mínimo de 0 que es su valor BOTTOM, hasta un máximo que es su valor TOP.

Con la resolución de 8 bits, entonces su máximo será 255, luego al ir el registro TCNT2 desde 0 hasta 255 habrán transcurrido 255us luego volverá a 0, pero en esa vuelta a 0 transcurre 1us mas, por lo que en ir de 0 a 255 y volver a 0 han pasado 256us, que vendría a ser el periodo de la señal PWM timer1 AVRTpwm cuando la FCPU es de 1Mhz, no siempre se utiliza una FCPU de 1Mhz esto puede variar, y como consecuencia variará el tiempo que pase para que el registro TCNT2 aumente su valor desde 0 a 255 en este caso.

Seguidamente podemos apreciar la señal PWM que tenemos de nuestros motores:

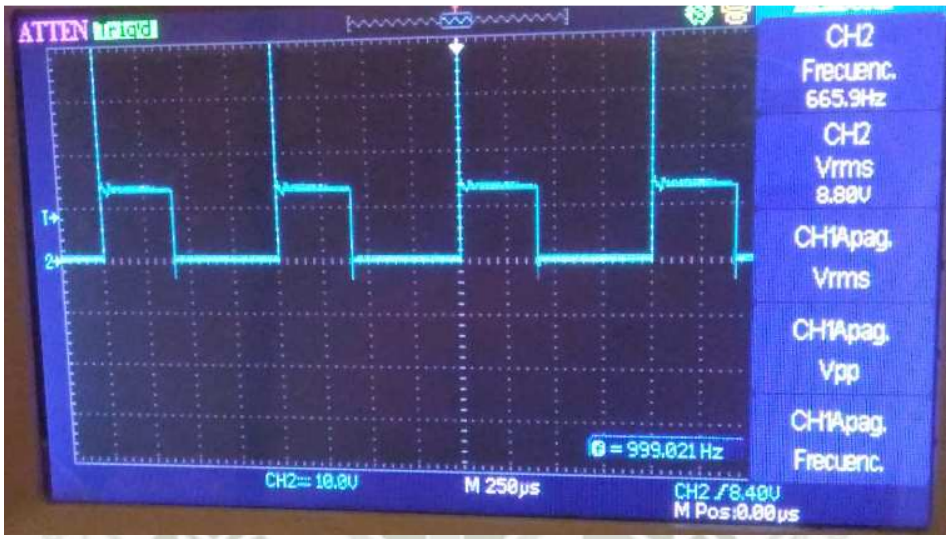


FIGURA 6.46. Variación del PWM a 8.80V.

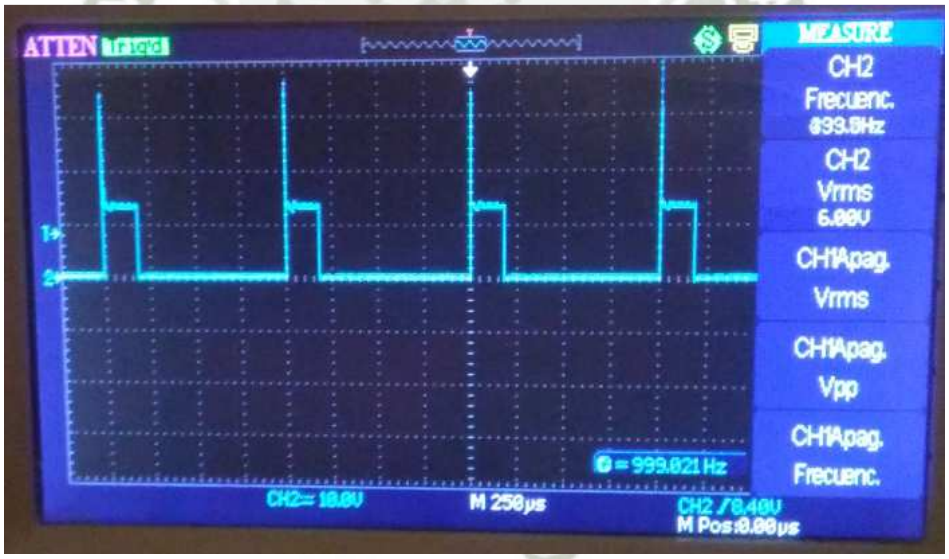


FIGURA 6.47. Variación del PWM a 6V.

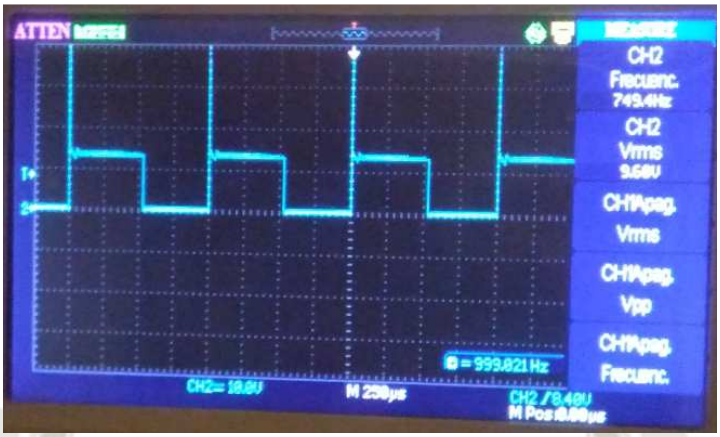


FIGURA 6.48. Variación del PWM a 9.60V

6.10.3. Módulos XBEE

Para la conexión de los módulos XBEE con el microcontrolador, primeramente es necesario conocer todos los pines y la función de cada uno de ellos. Los pines pueden verse en la siguiente imagen:

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

FIGURA 6.49. Tabla de pines de los módulos XBEE[11]

[11] Imagen obtenida del datasheet del microcontrolador 1284P

Algunas de las conexiones que deben hacerse mínimamente dependiendo de la acción que quiera realizarse para diseño son las siguientes:

- Conexiones mínimas: Vcc, GND, Dout y Din.
- Conexiones mínimas para la actualización de firmware: Vcc, GND, Din, Dout, ,RTS y DTR.
- La dirección de la señal es especificada con respecto al módulo.
- El módulo incluye una resistencia pull-up de 50KΩ, adjunta para RESET (activación en bajo L).
- Varias de las entradas pull-ups pueden ser configuradas utilizando el comando PR.
- Los pines libres deben dejarse desconectados.

Ahora para comenzar con la circuitería del XBEE mostraremos la siguiente figura, en donde se muestra las las conexiones mínimas que necesita el módulo para poder ser utilizado. Después de estas mínimas conexiones, se debe configurar según el modo de operación adecuado para la aplicación requerida.

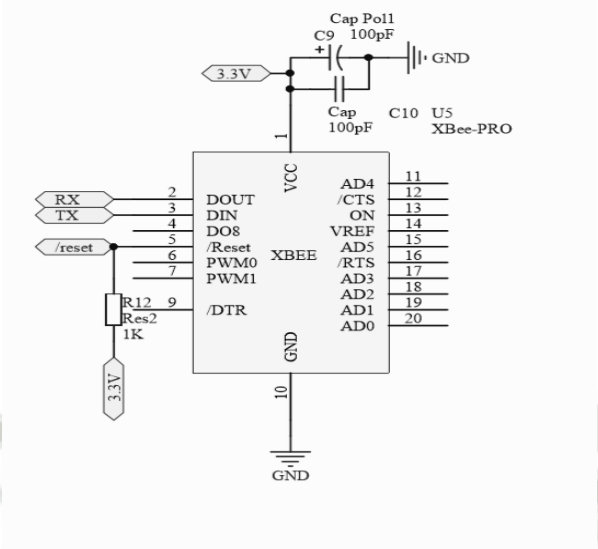


FIGURA 6.50. Esquemático del módulo XBEE

El módulo requiere una alimentación desde 2.8 a 3.4 V, en este caso 3.3V, A continuación se muestran los diagramas electrónicos entre el microcontrolador y el módulo XBEE. El regulador utilizado en este caso es el LM1117, el cual es un componente que se consigue fácilmente. Para trabajar con el módulo XBEE es necesario el divisor de voltaje a la salida del microcontrolador

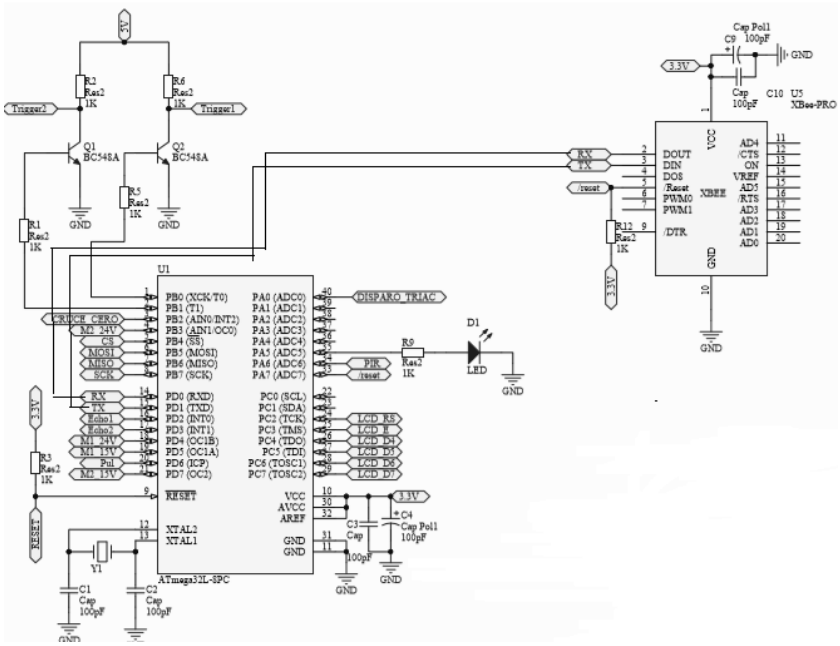


FIGURA 6.51. Esquemático de los pines utilizados en el módulo XBEE.

El módulo XBEE debido a sus características de alimentación se está utilizando un regulador de voltaje, a 3.3V, al igual que el microcontrolador de manera similar trabaja el control de mando.

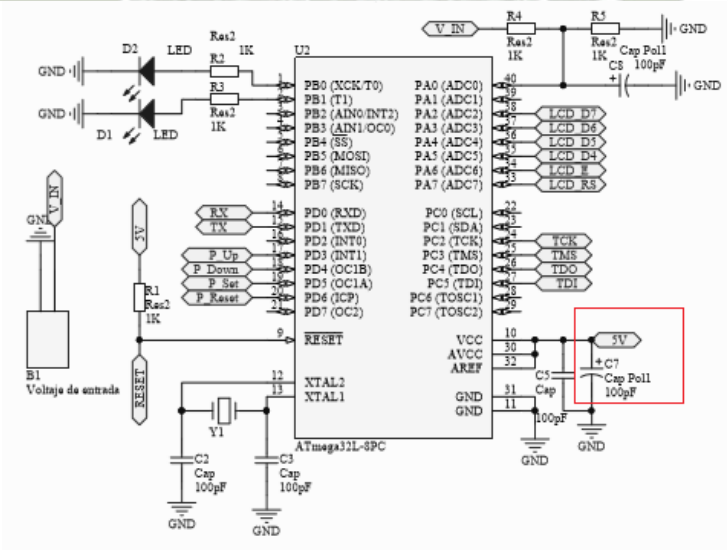


FIGURA 6.52. Esquemático del microcontrolador con referencia al regulador de 3.3V utilizado

Por último lo que es configuración del propio módulo XBEE se vio anteriormente en el apartado de configuración de módulos XBEE.

6.10.4. Sensores ultrasónicos

Para utilizar estos sensores, lo primero que debemos tener en cuenta es que en el microcontrolador para poder activar los ultrasónicos, se usa una configuración de conversor lógico de 3.3V a 5V; una vez encendidos los ultrasónicos, comienzan a usar el cronómetro (un solo timer/counter) solo se usa uno para los dos sensores de forma alternada, y el temporizador es detenido por la señal de retorno (echo).

Como se mencionó anteriormente, veremos el adaptador lógico para los ultrasónicos:

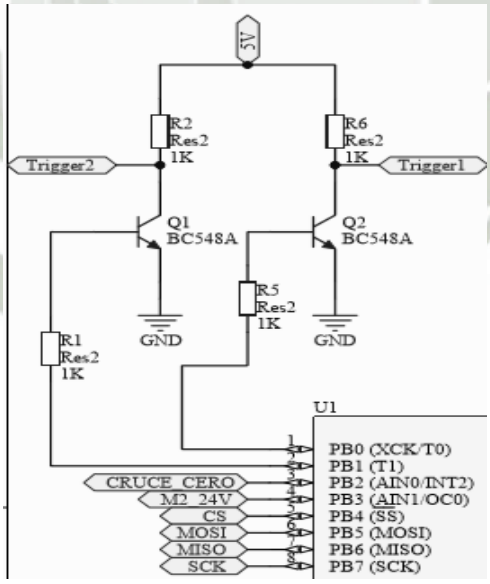


FIGURA 6.53. Adaptador lógico en el microcontrolador para los sensores ultrasónicos

En esta parte de la configuración del microcontrolador haremos el cálculo de Rb en los dos transistores para un conversor de 3.3V a 5V.

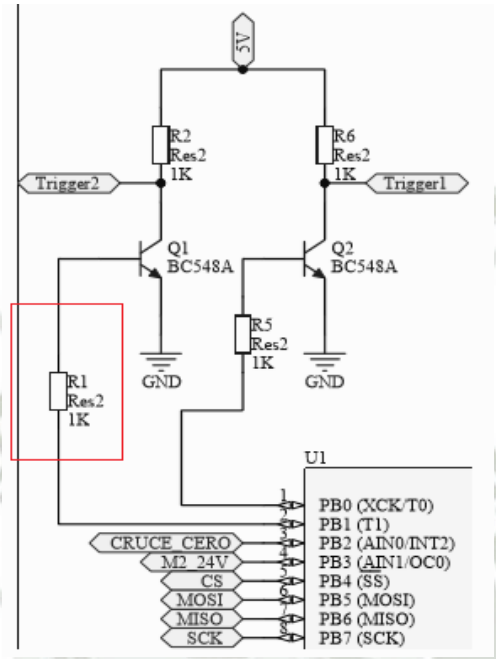


FIGURA 6.54. Esquemático del adaptador lógico para hallar Rb

Para el cálculo de Rb se sabe que:

$$Vbb = Ib\ Rb - Vbe$$

Pero también sabemos que para su máximo rango de Ic:

Si: B = 110 e Ic = 100mA

$$Ic = B\ Ib$$

$$Ib = 0.9mA$$

Entonces:

$$3.3V = (0.9mA)(Rb) - 0.7$$

$$2.6 = (0.9mA)Rb$$

$$Rb = 2.9K \rightarrow 2.7K \text{ valor comercial}$$

Hallamos Rb con 100mA como referencia por ser la máxima corriente permitida en Ic para su óptimo funcionamiento.

Sin embargo, al trabajar en colector, este se comporta como una fuente de corriente para poder trabajar con el voltaje requerido en los ultrasónicos.

Entonces para hallar Rc:

$$Vc = Ic \times Rc$$

Donde para obtener 5V. eficientemente, Ic = 5mA

$$5V. = 5mA \times Rc$$

$$Rc = 1K$$

Para demostrar que al pasar los 100mA en Ic, el transistor se dañaría, tenemos el siguiente ejemplo:

Siendo Vcc = 10V con una Rc = 100 Ω

$$Vcc = Ic \times Rc$$

$$10V = Ic \times 100$$

$$Ic = 100mA$$

Dónde:

$$Ic = B\ Ib$$

$$100mA = 110\ Ib$$

$$Ib = 0.9mA$$

El cual estaría en el tope del rango de corriente; sin embargo al elevar el voltaje a 12V tendremos que:

$$12V = Ic\ (100)$$

$$Ic = 120mA$$

Y:

$$Ic = B\ Ib$$

$$120 = 110\ Ib$$

$$Ib = 1.09mA$$

Esto ocasionaría en la malla de Vc, al ser limitado Ic :

$$Vcc = Vc + Vce$$

$$Vcc = IcRc + Vce$$

$$12V = (100mA)(10V) + Vce$$

$$12V = 10V + Vce$$

$$Vce = 2V$$

Donde en la configuración dada anteriormente:

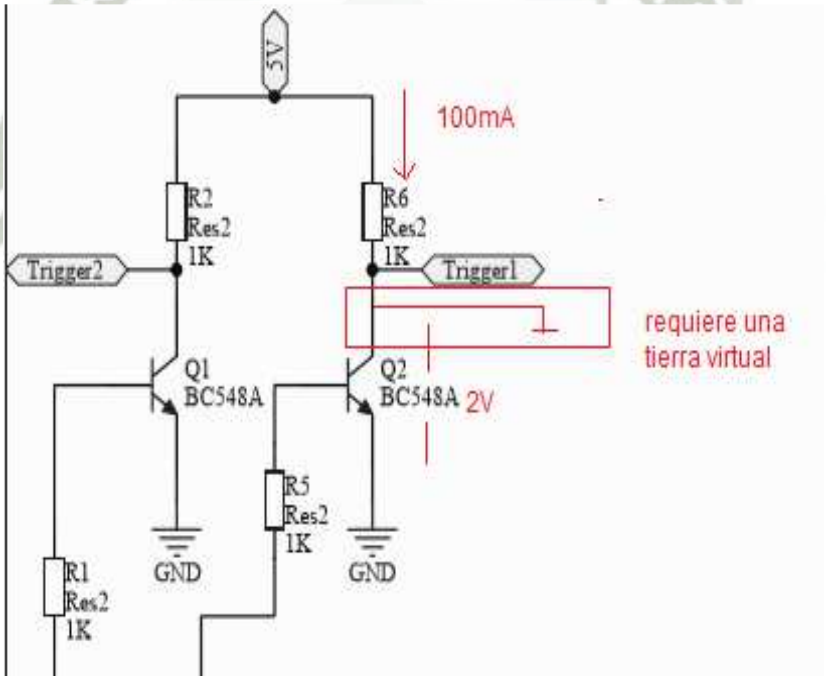


FIGURA 6.55. Aparición de tierra virtual

Entonces al existir un voltaje en Vce = 2V nuestra tierra virtual desaparece por lo tanto ya no se tiene la configuración que necesitamos y los sensores ultrasónicos ya no son alimentados.

6.11. DISEÑO DEL SOFTWARE

El diseño del siguiente programa se realizó mediante el lenguaje C, se optó por este lenguaje de programación ya que es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo" (permitiendo ciertas licencias de ruptura).

Como es sabido, hacer una programación no es tan sencillo, por lo que el programa entero por ser bastante amplio se encuentra en el **Anexo 2.**

A continuación se muestra el diagrama de flujo de la placa maestra, para así poder entender mejor el funcionamiento general del programa.

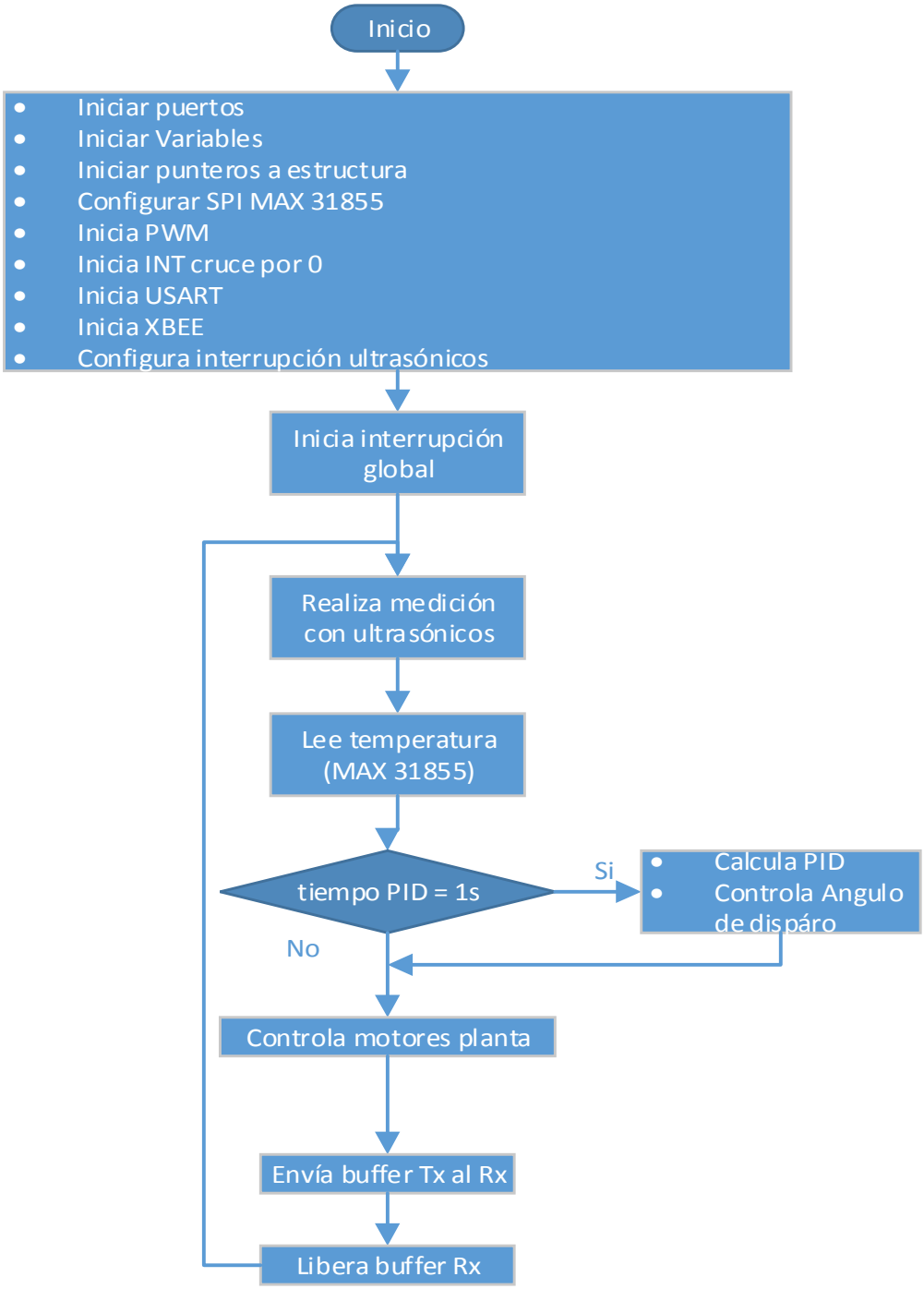


FIGURA 6.56. Diagrama de control general

Después de ver el funcionamiento global del programa en el apartado anterior, el programa llama a interrupciones para que estos realicen acciones específicas en la programación, estas interrupciones son las que se muestran a continuación:

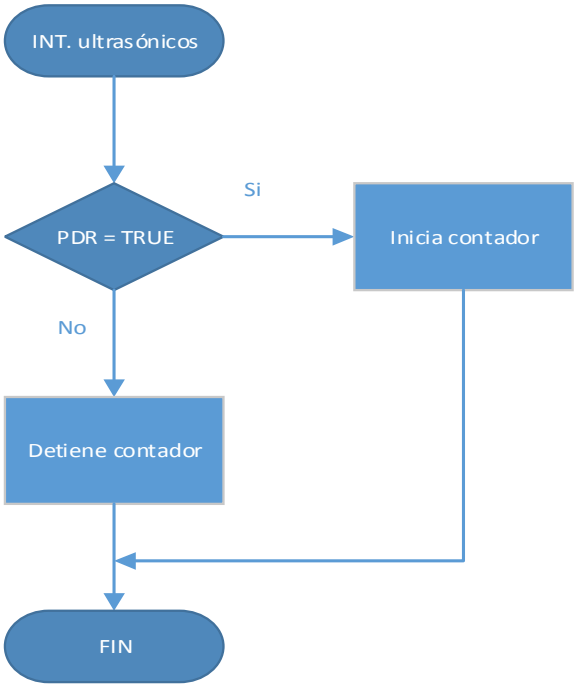


FIGURA 6.57. Interrupción de sensores ultrasónicos

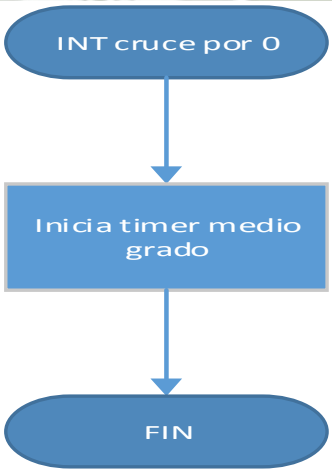


FIGURA 6.58. Interrupción de cruce por cero

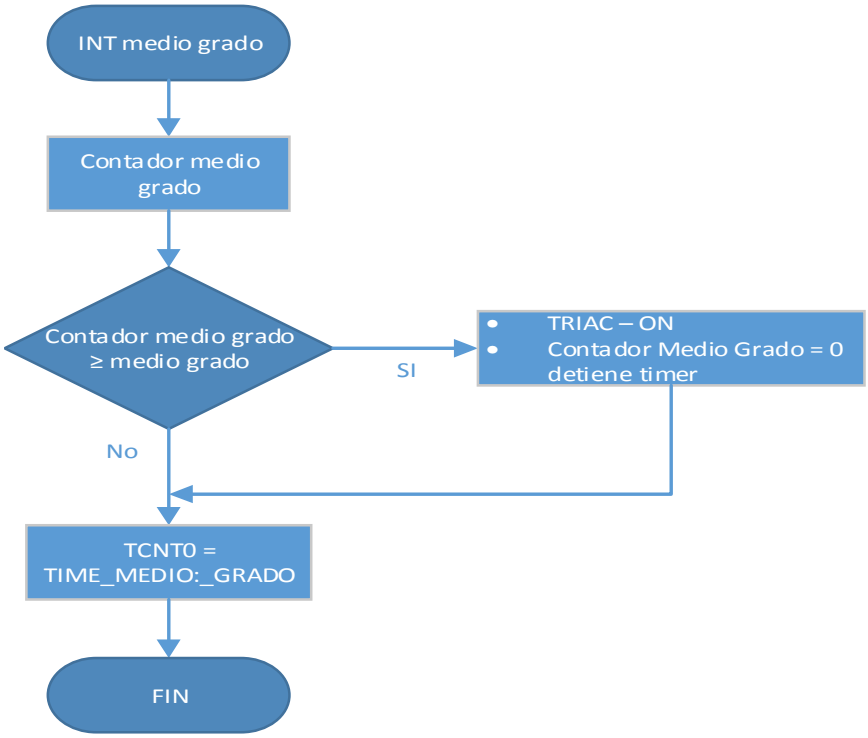


FIGURA 6.59. Interrupción medio grado

Seguidamente se presentará el funcionamiento general del programa en el control de mando. Mediante este control es posible la automatización por radiofrecuencia; es decir, es el componente principal, ya que este es el que alimenta la toma las decisiones, dependiendo de los parámetros medidos y monitoreados, ya vistos en apartados anteriores. A continuación se muestra el diagrama de flujo del control de mando.

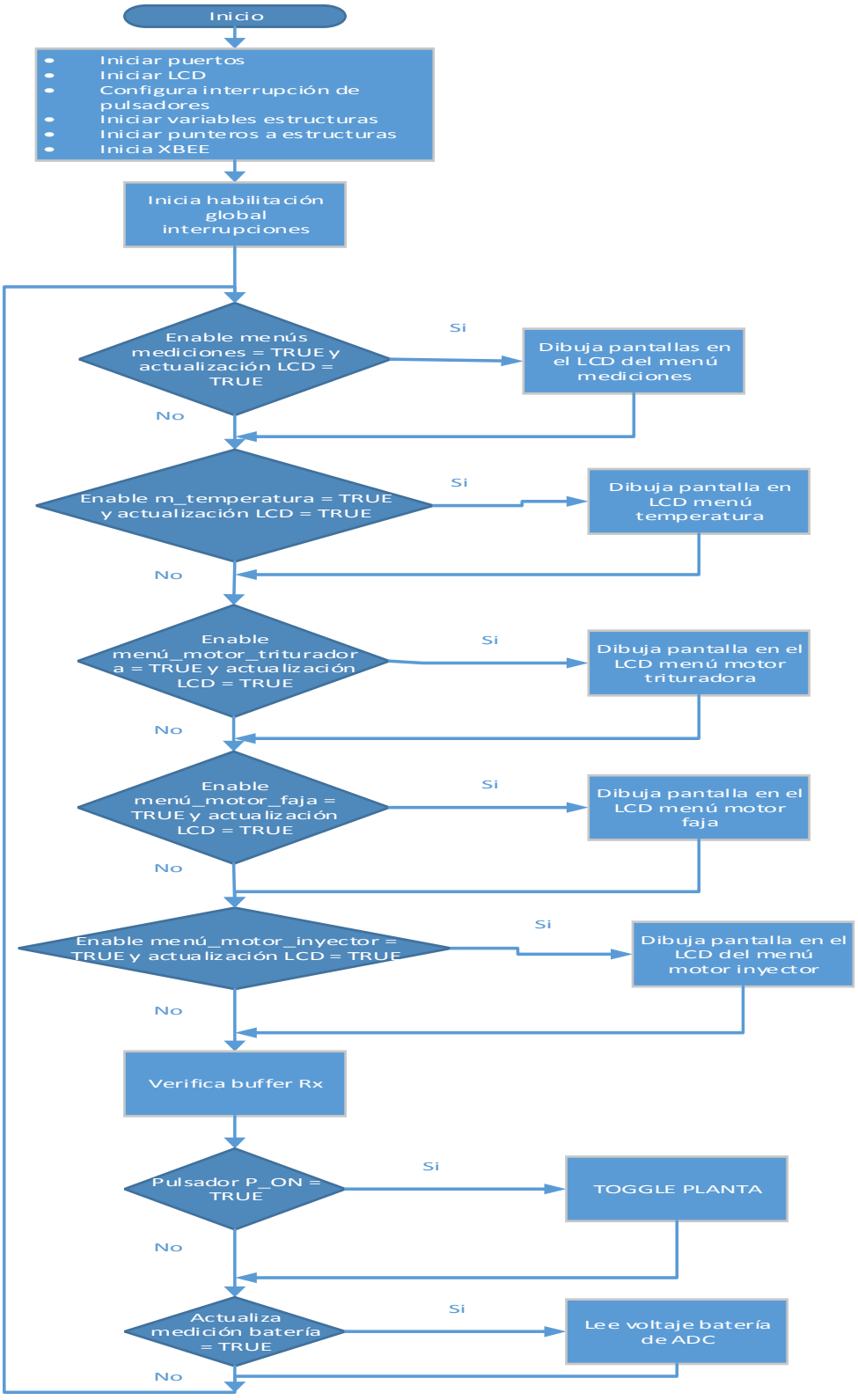


FIGURA 6.60. Digrama de flujo del control de mando

Como ya mencionamos anteriormente existen interrupciones dentro del programa, sin embargo para este caso solo contaremos con una, la que se muestra a continuación:

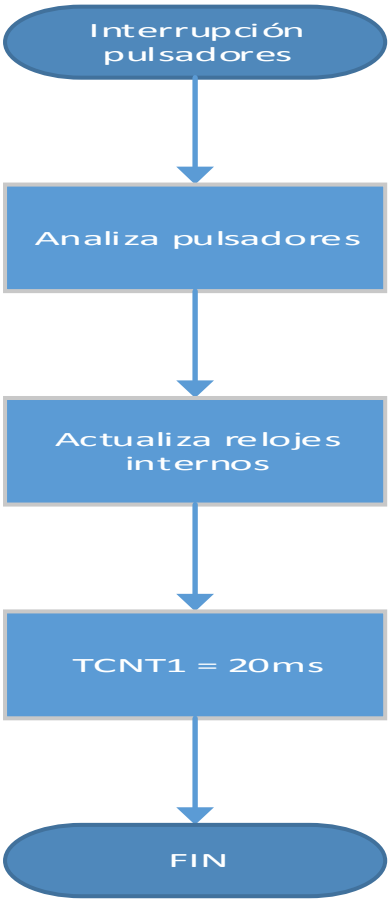


FIGURA 6.61. Interrupcion de pulsadores

CAPITULO VII: DESARROLLO DEL PROYECTO

Para poder automatizar la planta en el modelo anteriormente presentado y en el apartado 6.1, es necesario contar con la parte electrónica la cual consta de sistemas de circuitos para que el proceso anterior pueda ser controlado y monitoreado si no es en su totalidad, en la gran mayoría del proceso.

7.1. COMPONENTES UTILIZADOS

7.1.1. Componentes Electrónicos

a) **PLACA MAESTRA (Transmisor):** compuesta por los dispositivos:

- Microcontrolador Atmega 1284P
- LM2576.
- LM1117
- IR2110.
- MOSFET IRfz48n
- Max31855
- CD40106BE
- MOC4031
- 4N35
- Diodo B36
- Bobina 101
- Módulo XBEE
- xtal 11.0592
- diodos

- fusibles
- leds
- capacitores varios
- resistencias varias
- disipador

b) PLACA DE MANDO (Receptor), compuesta por:

- Microcontrolador Atmega 1284P
- LM2576.
- Módulo XBEE
- Diodo B36
- Bobina 101
- Capacitores varios
- Resistencia varias
- Led rojo
- Pantalla LCD
- Pulsadores

7.1.2. Sensores.

Corresponden a los sensores necesarios para aplicación en el modelo demostrativo

- 1 Sensor PIR de movimiento
- 2 Sensores ultrasónico HC-SR04
- 1 sensor de temperatura.

7.1.3. Componentes Mecánico-Eléctricos

- Motores de 12 a 24 Vdc.
- Fuentes de alimentación de 12 y 24V. regulables

7.2. CIRCUITOS Y DISPOSITIVOS APLICADOS A LA AUTOMATIZACIÓN

Para poder desarrollar correctamente los circuitos que serán usados en la automatización del proceso de reciclaje de plástico, se tiene que reconocer una de las partes más importantes que son los circuitos integrados ya que en función a éstos, de acuerdo al diseño que se les dé, se automatizará la planta.

7.2.1. Circuito de Placa Maestra

Primeramente se presentará la placa entera del circuito de planta, el cual lo apreciamos a continuación.

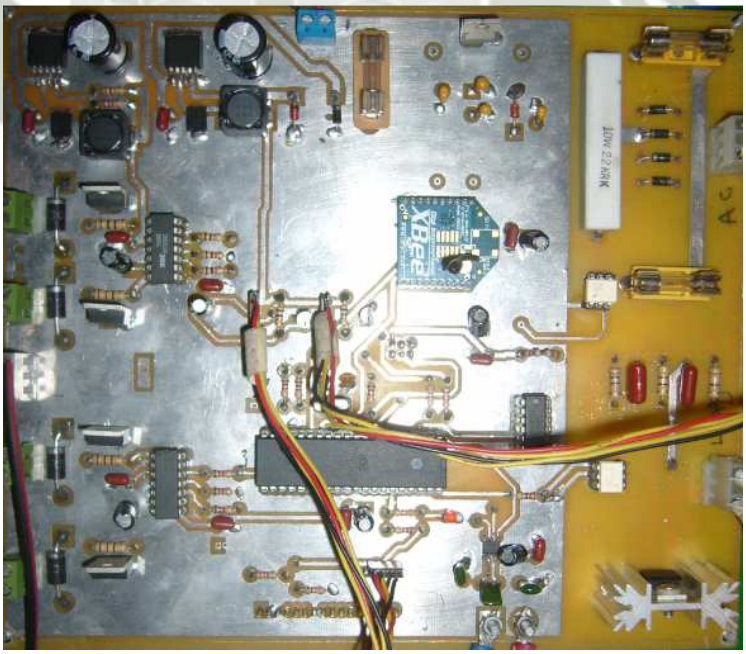


FIGURA 7.1. Circuito General en placa

a) Circuito 1: Regulador de Voltaje

La función que cumple en este caso es transformar un voltaje de entrada a un voltaje con el que necesitemos trabajar ya sea 3,3V, 5V a más, dependiendo de la capacidad del regulador. Los utilizados en este circuito son LM2576 y el LM1117,este último se cambió debido al problema de trasientes que se tuvo anteriormente; podemos apreciar el integrado LM2576 en la siguiente imagen (ver figura 7.2.).

FIGURA 7.2. Integrado LM2576

En este caso se está usando 2 reguladores de voltaje ADJ de alta eficiencia, ya que necesitamos trabajar con salidas de voltajes de 15V y 5V. Se escogió el integrado LM2576, por las características que muestra; es un regulador de voltaje ajustable ADJ de alta eficiencia. El LM2576 de los reguladores son circuitos integrados monolíticos que ofrecen todas las funciones activas para un paso-bajo, (buck) regulador de conmutación, capaces de conducir la carga 3A con excelente línea y regulación de carga. Estos dispositivos están disponibles en tensiones de salidas fijos de 3,3V, 5V, 15V, 24V y una versión ajustable de salida. El rango de la versión ajustable de salida es de 1.23V a 37V.

Además de un LM1117 el regulador del cual necesitamos 3.3V, como ya se habló debido al problema de trasientes, y porque los componentes que trabajan a este voltaje son muy sensibles a subidas de voltaje y son los de mayor costo. A continuación podemos apreciar el integrado LM1117:

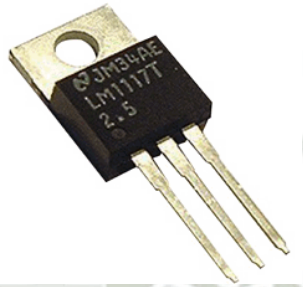


FIGURA 7.3. Integrado LM1117

Requieren un número mínimo de componentes externos; estos reguladores son simples de usar, incluyen una compensación de frecuencia interna y un oscilador de frecuencia fija. Este regulador ofrece una eficiencia alta para remplazar populares reguladores lineales de tres terminales. Sustancialmente reduce el tamaño del disipador de calor, y en algunos casos no requiere de disipador de calor, además de ser mucho más veloz que el LM2576. El esquema circuital y el diagrama interno del integrado LM2576 se muestra en la siguiente figura:

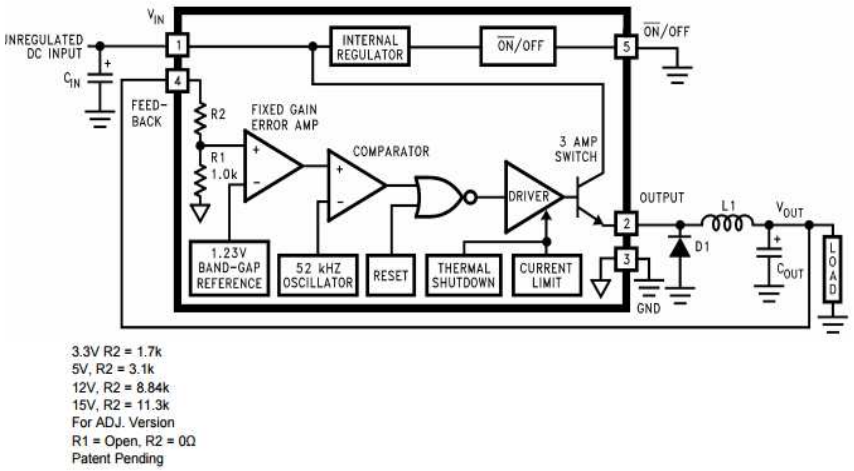


FIGURA 7.4. Esquema circuital y diagrama interno del circuito integrado LM2576

Para trabajar con el voltaje de 3.3V, se escogió el integrado LM1117, debido que este es un regulador mucho más rápido, esto debido al problema de trasientes que se tuvo antes, y daño varios componentes entre ellos el módulo de transmisión XBEE, se tuvo que cambiar por cuestiones de seguridad de otros componentes el LM2576 por el LM1117 para los componentes que usan 3.3V para su funcionamiento.

Finalmente, apreciaremos los circuitos ya armados en placa, a continuación:

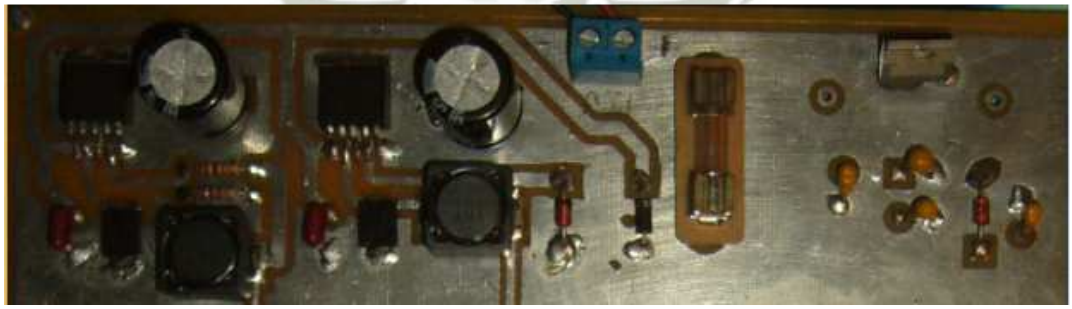


FIGURA 7.5. Circuito en placa de los reguladores de voltaje

b) Circuito 2: Driver de disparo

El siguiente circuito a describir es el driver; este dispositivo nos sirve para poder disparar un MOSFET que veremos en el esquemático, amplifica voltaje y corriente mayor en la salida de acuerdo a lo requerido por el MOSFET. Funciona como switching al disparar el MOSFET como control on/off. Podemos apreciar el CI usado en la siguiente figura.

FIGURA 7.6. Drivers de disparo IR2110

En este caso estamos usando para el driver el integrado IR2110, de alto voltaje, y un MOSFET de potencia de alta velocidad. Sus principales características son: canal flotante diseñado para la operación de arranque en pleno funcionamiento a +500V, tiene un rango de alimentación en la unidad de puerta (gate) de 10 a 20V, bloqueo de mínima tensión para ambos canales, compatible con la lógica de 3.3V, rango de suministro lógico separado de 3.3V a 20V, lógica y poder de tierra de ± 5 voltios compensado; cuenta con dos entradas y dos salidas, además de una entrada tipo digital cada entrada está destinada a una salida; lo cual se puede apreciar en el diagrama interno. En este caso estamos usando dos drivers los cuales podemos apreciar en el esquemático del circuito del inversor en la siguiente figura (ver figura 7.7.).

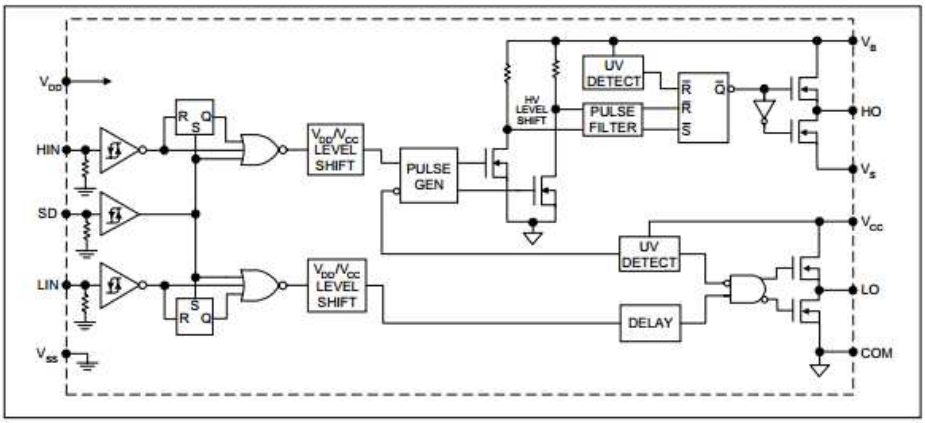


FIGURA 7.7. Diagrama esquemático del circuito integrado IR2110

Este IR2110 cuenta con una entrada de alimentación digital, en este caso 3.3V, y está aislado a la parte de potencia. Este integrado cuenta con una configuración de fuente flotante que es muy potente, pero en este caso no la estamos usando.

Una de sus características más importantes y más resaltantes es que éste integrado tiene la posibilidad de trabajar con fuente flotante; es decir que aislando la parte digital que son los 3.3V con los que trabaja y las entradas Hin y Lin también de 3.3V, la parte de potencia puede trabajar como su datasheet lo dice hasta 500V en el caso de IR2110 y 600 con el IR2113, y resiste hasta 2A de corriente aislando las tierras tanto de las entradas como de las salidas y el Vs al sourge del MOSFET, o sea que se puede utilizar este tipo de circuito con componentes un poco más potentes y mover un motor de bastante potencia sin problema. Sin embargo como ya mencionamos anteriormente en este caso no estamos usando fuente flotante debido a que el voltaje a usar es 15V y

el driver no tiene problema alguno para trabajar con este voltaje; se podría decir que es bajo para la capacidad que éste tiene.

Este integrado lo estamos usando de forma tal que al ingresar el voltaje por las entradas Hin y Lin es de 3.3V., tras la configuración de dicho integrado, este pueda disparar el MOSFET para que podamos conseguir el voltaje deseado y necesario que, en este caso, es de 12V. a 24V. y lo trabaja con PWM para el funcionamiento de los motores. Tanto en el primero como en el segundo integrado sirven para más de dos motores, en el proyecto que se está trabajando.

Luego tenemos el **MOSFET IRfz48n**. Como sabemos el MOSFET es un dispositivo controlado por tensión; es un dispositivo extremadamente veloz en virtud a la pequeña corriente necesaria para estrangular o liberar el canal. Por esta facultad se los usa ampliamente en conmutación. Su velocidad permite diseñar etapas con grandes anchos de banda minimizando, así, lo que se denomina distorsión por fase.

Las características más importantes de este MOSFET son: el R_{dson} que en este caso es igual a 14mΩ y que varía según la temperatura, tiene un zener de protección, soporta hasta 64A.

Además que trabaja como un interruptor minúsculo; sabemos que si en un MOSFET la tensión entre la Puerta y la Fuente es menor que la tensión umbral, Vgs<Vt, el transistor está cortado. Es decir, entre los terminales de Fuente y Drenador, la corriente es nula, ya que existe un circuito abierto. Sin embargo, cuando Vgs es mayor que Vt se crea el

canal, y el transistor entra en conducción. Cuanto mayor es la tensión de puerta menor es la resistencia del canal, y ésta puede llegar a aproximarse a un cortocircuito. Así, el MOSFET es capaz de funcionar como un interruptor.

A continuación se muestra el circuito, ya armado en placa el cual estamos utilizando:

FIGURA 7.8. Circuito en placa para el control de motores

c) Circuito 3: control del Calefactor en el Extrusor

Lo que principalmente se hace en este circuito es trabajar específicamente con el extrusor. El circuito está diseñado para que la temperatura que varía en el extrusor pueda ser automatizada gracias al PID que será controlado mediante el programa de automatización en el microcontrolador. (en donde todo el diseño matemático y del PID se hizo en el capítulo anterior). Aquí podemos apreciar la parte del circuito del control de temperatura:

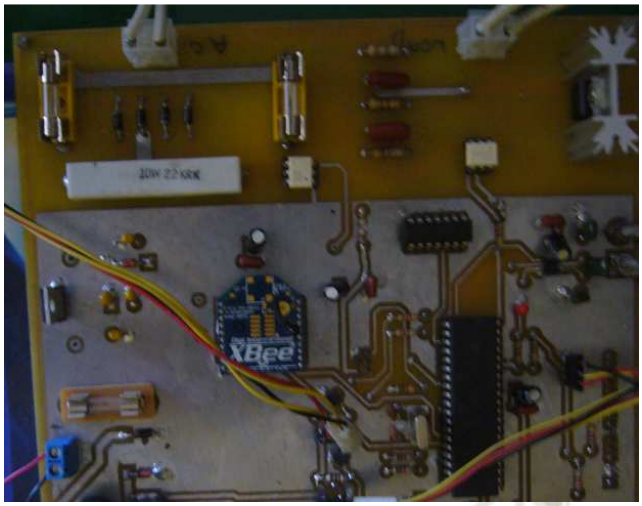


FIGURA 7.9. Circuito en placa del control de temperatura

Sin embargo otro punto importante de este circuito es el diseño de seguridad para trabajar con el extrusor, aislando la corriente AC de la otra parte del circuito; ya que dicho extrusor trabaja con corriente AC, y es por ello que se opta por este circuito para poder dar la seguridad correspondiente a toda la placa. El circuito antes mencionado es el siguiente:

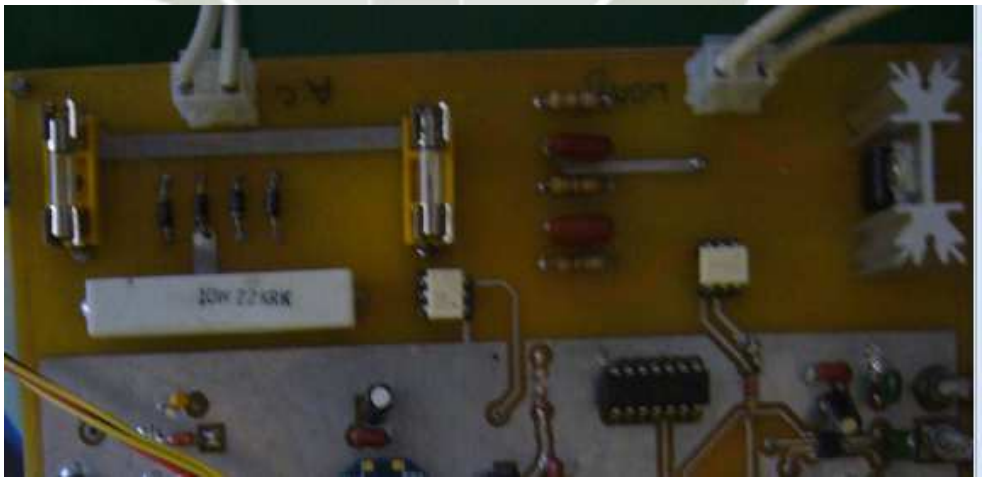


FIGURA 7.10. Circuito en placa para el control de temperatura

Como podemos apreciar en la parte superior primeramente lo que hace este circuito es rectificar la onda AC con un puente de diodos para que esta se convierta en DC. Seguidamente después del puente de diodos en la parte superior del circuito se aprecia el optoacoplador; se utiliza este dispositivo a modo de interfaz, de tal forma que esa parte previa del circuito antes visto, quedarían unidos ópticamente a la placa, lo que a efectos da protección del circuito, se traduce en colocar una resistencia de un valor muy alto (muchos MΩ), lo que lo hace especialmente útil para proteger contra los picos de tensión, ya que por razones obvias el trabajar con 220Vac. es un riesgo sea cual sea la forma de trabajo. Ahora encontramos el CD4016B que es un inversor séxtuple de disparo Schmitt en donde un circuito funciona como un inversor de la señal y el otro sirve para mejorar la calidad de la misma señal.

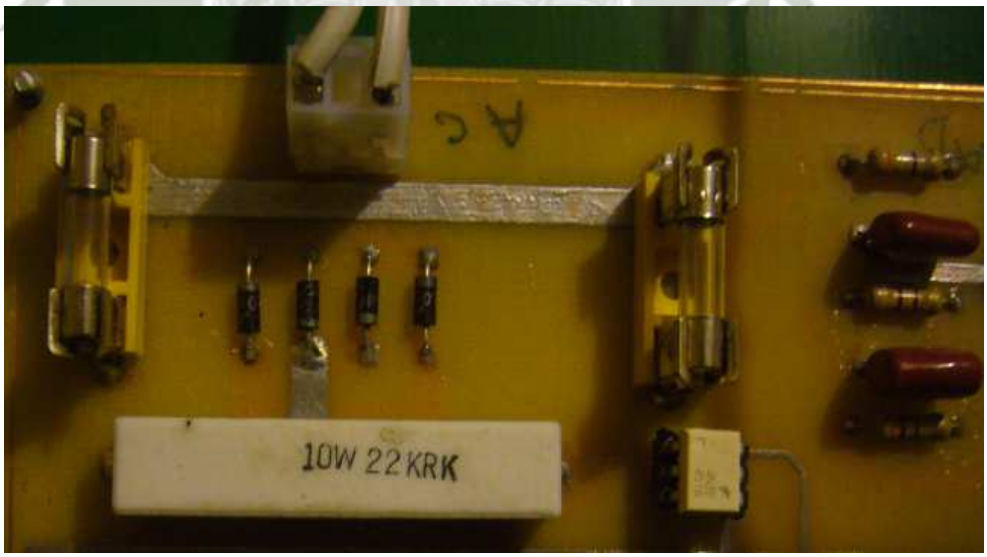


FIGURA 7.11. Circuito en placa del control de cruce por cero

Ahora en la parte inferior del circuito podemos ver un arreglo con un triac y un optotriac, en donde gracias a la versatilidad del TRIAC y la simplicidad de su uso lo hace ideal para una amplia variedad de aplicaciones relacionadas con el control de corrientes alternas. Una de ellas es su utilización como interruptor estático ofreciendo muchas ventajas sobre los interruptores mecánicos convencionales, que requieren siempre el movimiento de un contacto, siendo la principal la que se obtiene como consecuencia de que el TRIAC siempre se dispara cada medio ciclo cuando la corriente pasa por cero, con lo que se evitan los arcos y sobre tensiones derivadas de la conmutación de cargas inductivas que almacenan una determinada energía durante su funcionamiento, para luego de dispararse, llegar a un optotriac, que al no existir conexión eléctrica entre la entrada y la salida, el acoplo es unidireccional (LED al foto-TRIAC) y permite un aislamiento eléctrico entre ambos dispositivos de hasta 7500 V. Además, algunos foto-TRIAC incluyen un circuito de detección de paso por cero que permite sincronizar señales de la red eléctrica con señales de control del LED para ajustar el ángulo de conducción. Como es el caso del moc3021 que únicamente requiere 10 mA de corriente en el LED; cuando el LED está apagado, el foto-TRIAC está bloqueado conduciendo una pequeña corriente de fuga denominada I_{drm}, cuando el diodo conduce, dispara el foto-TRIAC circulando de esta forma entre 100 mA y 1A. Al no ser un dispositivo que soporte grandes niveles de potencia, el propio optotriac actúa sobre el control del TRIAC.

Se muestra el circuito mencionado anteriormente, armado en placa listo para su funcionamiento:

FIGURA 7.12. Circuito en placa del control de disparo del triac

d) Chip Max31855: chip para el sensor de temperatura

Este chip básicamente lo que hace es poder amplificar el micro o mini voltaje del sensor, y enviar una salida digitalizada. Trabaja con compensación de unión fría y digitaliza la señal de termocuplas tipo K-, J-, N-, T-, S-, R- O E-. Emite dato de salida en una señal de 14-bits, SPI-compatible, formato de solo lectura. Este convertidor resuelve temperaturas a 0,25°C, permite temperaturas de hasta +1800°C y tan bajas como -270°C, detecta cortos a GND o Vcc y detecta termocupla abierta.

Uno de los puntos más importantes de este chip es que simplifica todo un arreglo de operacionales para el sensor de temperatura, esto lo hace bastante eficiente al momento de diseñar. A continuación se explica cómo va trabajando dicho chip: el sensor en este caso de tipo J

convierte la temperatura en pequeños mini voltios dependiendo del grado de la temperatura, envía una señal tanto positiva como negativa. Como se puede apreciar dentro del diagrama esquemático del integrado, se tiene un detector de fallas, que es el que hace que sepamos si existe un corto con GND, Vcc o que este abierto, al no haber fallas, pasa a un amplificador de voltaje, después de ser amplificado el voltaje se junta con lo que se llama compensación de unión fría, la cual es su propio sensor de temperatura interno el que, como su nombre lo dice, compensa la temperatura ambiente donde se encuentra la termocupla, esto nos sirve para que la medición sea mucho más precisa; luego pasa a un convertidor análogo digital y también cuenta con su reloj para poder tener una comunicación síncrona con el microcontrolador a donde irá la señal. (ver figura 7.13.)

FIGURA 7.13. Chip Max31855 para sensor de temperatura

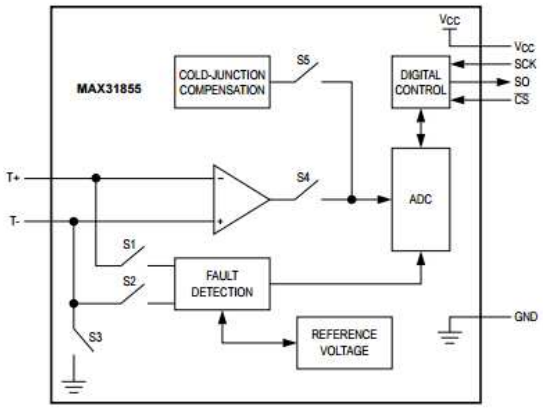


FIGURA 7.14. Diagrama del chip sensor de temperatura

A continuación podemos ver una tabla de los tipos de sensores de temperatura que existen, su rango de medición y los materiales de lo que están hechos, además del tipo J que estamos usando que es lo suficientemente apropiado para nuestra aplicación.

TYPE	T- WIRE	T+ WIRE	TEMP RANGE (°C)	SENSITIVITY (µV/°C)	COLD-JUNCTION SENSITIVITY (µV/°C) (0°C TO +70°C)
K	Alumel	Chromel	-270 to +1372	41.276 (0°C to +1000°C)	40.73
J	Constantan	Iron	-210 to +1200	57.953 (0°C to +750°C)	52.136
N	Nisil	Nicrosil	-270 to + 1300	36.256 (0°C to +1000°C)	27.171
S	Platinum	Platinum/Rhodium	-50 to +1768	9.587 (0°C to +1000°C)	6.181
T	Constantan	Copper	-270 to +400	52.18 (0°C to +400°C)	41.56
E	Constantan	Chromel	-270 to +1000	76.373 (0°C to +1000°C)	44.123
R	Platinum	Platinum/Rhodium	-50 to +1768	10.506 (0°C to +1000°C)	6.158

TABLA 7.1. Tipos y Características de los Chips Sensores de Temperatura

Ahora podemos ver el circuito armado para nuestro chip **Max31855,** el cual usaremos en el sensado de la temperatura:

FIGURA 7.15. Circuito en placa para el chip MAX31855

e) El microcontrolador

Primero veremos lo que es AVR. Es una familia de microcontroladores del fabricante estadounidense Atmel. Esta familia es muy usada debido a su diseño simple y la facilidad de programación. El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profusamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador son usados como punteros de 16 bits al espacio de memoria externa, bajo los nombres X, Y y Z. El set de instrucciones AVR está implementado físicamente y disponible en el mercado en diferentes dispositivos, que comparten el mismo núcleo AVR pero tienen distintos periféricos y cantidades de RAM y ROM. La compatibilidad entre los distintos modelos es preservada en un grado razonable.

Los microcontroladores AVR tienen una cañería ('pipeline' en inglés) con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bits.

Veamos ahora el funcionamiento del atmega1284P. Este es un microcontrolador de 8 bits, de la familia atmega, 16 Mhz de instrucción por segundo, funciona al número del cristal con el que se trabaje, tiene un reloj interno (1, 4 y 8MHz); 128kbs de memoria flash de programa, 4K bytes EEPROM. Además cuenta con 16kbyte de SRAM interna, una de sus características más importantes es que soporta JTAG. ¿Qué es el JTAG?, el JTAG es una herramienta que se utiliza para probar submódulos de circuitos integrados y para depurar aplicaciones empotradas, o sea se puede visualizar paso por paso cada línea o aplicación de la programación en la computadora en diferentes programas dependiendo del lenguaje que se esté utilizando (limpia el código); es un depurador. Podemos ver el modelo del atmega1284P en la siguiente figura.

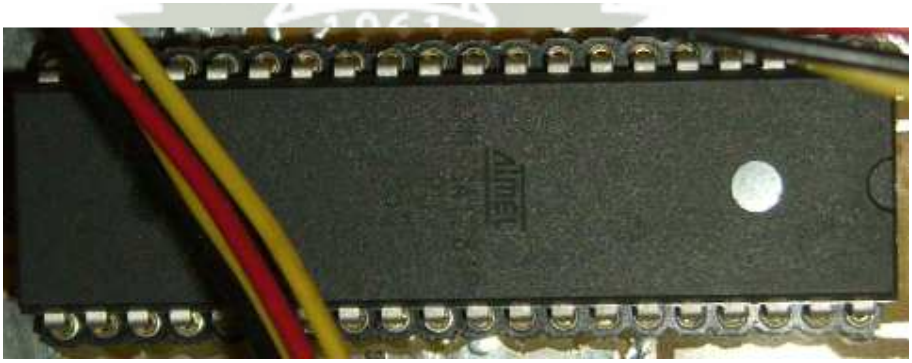


FIGURA 7.16. Micocontrolador atmega modelo 1284P

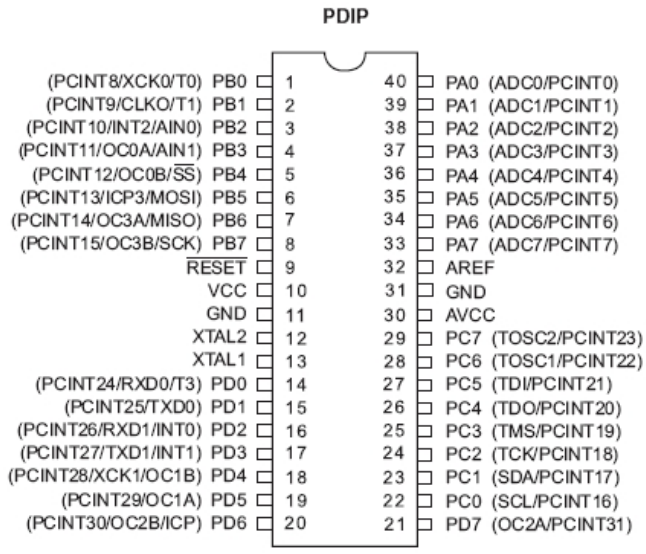


FIGURA 7.17. Diagrama de los pines del microcontrolador 1284P

Este modelo Cuenta con 4 timer/counter (dos de 8 bits y 2 de 16 bits), de los cuales se está usando uno de 16bits para los sensores de ultrasonido. El timer tiene varias fuentes posibles de interrupción. Dentro de la base de tiempos se puede generar una interrupción cuando se produzca un evento de actualización, que es lo que se está haciendo.

El timer, es utilizado obviamente para medir tiempo, en un AVR sirve para hacer tareas de forma asíncrona al programa principal. Esto puede pensarse así debido a que un Timer funciona de forma independiente al núcleo de AVR. Los timers pueden configurarse para producir una salida en un predeterminado pin reduciendo la carga del microcontrolador. Un timer debe tener un reloj asociado que le diga el paso, cada pulso incrementa el timer en uno, el timer mide los intervalos en periodos de la siguiente forma:

$$Resolución = \frac{1}{frecuencia}$$

Esto significa que el menor tiempo posible que el timer puede medir es un periodo de la señal de reloj de entrada. Por ejemplo, si alimentamos al timer con una señal de 100Hz, el periodo sería:

$$Resolución = \frac{1}{100Hz} = 0.01s$$

El periodo sería de 0.01 segundos, de modo que nuestro timer mediría en múltiplos de este. Si medimos un retardo de 60 periodos de timer, entonces el retardo total sería 60 veces 0.01 segundos o 0.6 segundos.

En nuestro programa, existe una interrupción de desbordamiento de un timer/counter cada 20ms; está usando un contador en la pantalla para que se actualice cada 600ms, de la siguiente manera: si el reloj es ≥ 30 (20ms X 30 = 600ms) se actualiza la pantalla.

También cuenta con 8 canales ADC, los cuales como se sabe son convertidores analógico-digital, en este caso solamente estamos usando 1, el que se usa para la visualización de batería. Para cualquier conversión de ADC, debemos tener en cuenta estos tres pasos de cómo se realiza la conversión:

- Se debe tomar un punto de referencia, ya que no siempre el voltaje tomado como 0 V será 0; esto quiere decir que algunas formas de trabajo lo toman de manera diferente por ejemplo: 3.3v = 1024 binario y 1V = 310 binario.

- Se debe tener un voltaje máximo de conversión, ya que la resolución es inversamente proporcional al voltaje.
- Finalmente, tener en cuenta la frecuencia de muestreo.

El ADC o Conversor Analógico Digital es un módulo del microcontrolador que sirve para traducir una señal analógica a un valor digital. La señal analógica es un nivel de tensión cuyo origen puede ser desde un divisor de tensión hasta el más complicado transductor de señal.

La circuitería del ADC realiza la conversión por aproximaciones sucesivas dentro del rango de referencia de tensión y con una determinada resolución medida en bits.

Para la parte del PID se utilizó la siguiente ecuación para poder tener realción de proporcionalidad entre el % de potencia y el ángulo de disparo:

$$\% P = \frac{\pi - \alpha + \frac{sen2\alpha}{2}}{\pi} x100$$

Por otro lado cuenta con una interface serial USART; es el puerto serial del atmega1284P y este lo tiene como periférico, la gran mayoría de las familias de AVR poseen al menos una interfaz USART, la cual sirve para enviar y recibir datos seriales desde el microcontrolador a un dispositivo (una PC, por ejemplo) o a otro AVR, USART es normalmente relacionada con la especificación RS232, mientras que la comunicación USART establece niveles lógicos de 3 a 5V, en este proyecto lo estamos

usando como una comunicación asíncrona con los xbee, para ello se tiene que fijar el reloj es por eso que trabaja con 11.059MHz para que de acuerdo a sus características trabaje con un baud rate de 9600.

Tiene activada la interrupción por recepción de datos por USART, se usa el cristal 11.0592MHz porque de acuerdo a sus especificaciones al usar este cristal, y con el calibrado del UBRR en 71, se tiene un 0.0% de error. (Ver tabla 7.2.).

Table 19-11. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

TABLA 7.2. Calibrado de UBRR en razón al error

Envío de datos: Nuestro querido AVR va a comunicarse con el mundo exterior, para ello tenemos que escribir en el registro UDR un carácter ASCII (para que sea posible leerlo en pantalla), pero antes de escribir en UDR deberíamos saber que no hay un dato anterior enviándose, para ello está el registro de estado y control UCSRA, bit UDRE el cual se

pone en 1 cuando el buffer de transmisión está vacío (es decir, podemos transmitir).

Recepción de datos: Para recibir los datos transmitidos desde el Tx por el pin Rxd, vamos a leer el contenido de UDR, pero antes haremos un bucle de espera consultando al UCSRA, bit Rxc el cual se pone en 1 cuando el buffer de recepción está lleno.

También, tiene una interface serial de maestro/esclavo SPI, el cual es usado para el chip de temperatura, con una comunicación síncrona, cuenta con reloj y datos de entrada y salida.

Uno de los puntos más interesantes de este microcontrolador, es que cuenta con un arreglo interno para la seguridad del atmega, el cual podemos apreciar en la siguiente figura.

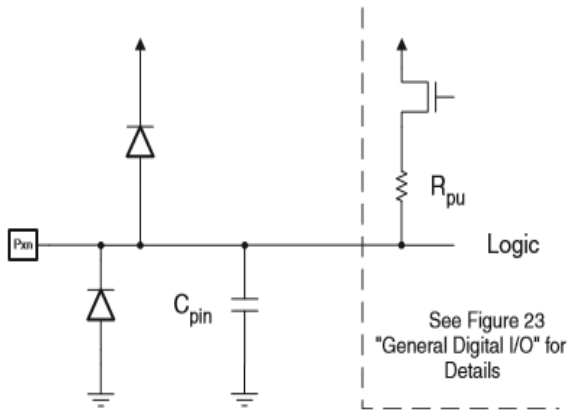


FIGURA 7.18. diagrama de arreglo interno del micocontrolador atmega modelo 1284P

Todas los pines I/O tienen diodos de protección para ambos Vcc y tierra como lo vemos anteriormente en la figura de arriba.

Una aplicación que tiene el timer/counter es el uso en los canales PWM, lo cual explicaremos esta forma de trabajo con PWM a continuación.

Además, la regulación por Ancho de Pulso de un motor de CC está basada en el hecho de que si se recorta la CC de alimentación en forma de una onda cuadrada, la energía que recibe el motor disminuirá de manera proporcional a la relación entre la parte alta (habilita corriente) y baja (cero corriente) del ciclo de la onda cuadrada. Controlando esta relación se logra variar la velocidad del motor de una manera bastante aceptable. En nuestro caso estamos aplicando para un motor para la faja transportadora en el modelo presentado, sin embargo la idea original es que se controlen dos motores, solo que en este caso no existe la trituradora ni el motor DC de esta parte del proceso, pero en un modelo industrial si habrían 2 motores o más los cuales serán controlados por PWM.

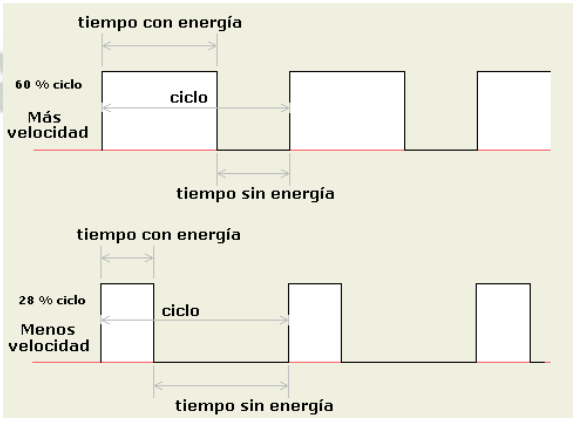


FIGURA 7.19. Variación de velocidad en relación al ancho de pulso

El microcontrolador no puede proporcionar la intensidad necesaria para mover un motor, es por ello que utilizamos algún tipo de driver en este caso el IR2110..

La función que más nos interesa en nuestro driver es:

- Capacidad para traducir nuestras señales digitales (del microcontrolador) a corrientes de mayor intensidad y voltajes mayores.

Se muestra el circuito armado para nuestro microcontrolador atmega 1284P:

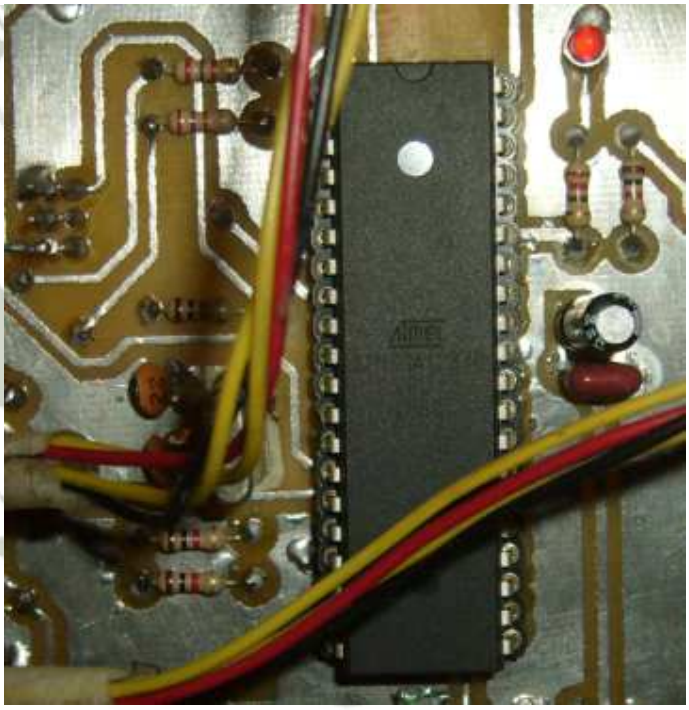


FIGURA 7.20. Circuito en placa del atmega 1284P

Configuración de modulos xbee con el microcontrolador:

El pin (OUT del XBee) va conectado al pin (Rx del microcontrolador) y el pin (IN del XBee) va conectado al pin (Tx del microcontrolador),además, **la comunicación entre el microcontrolador y el XBEE es directa, debido a que el microcontrolador ha sido alimentado con 3.3V**.

7.3. PLACA DE MANDO

Lo primero que se muestra es la placa de mando general, la cual podemos apreciar a continuación:

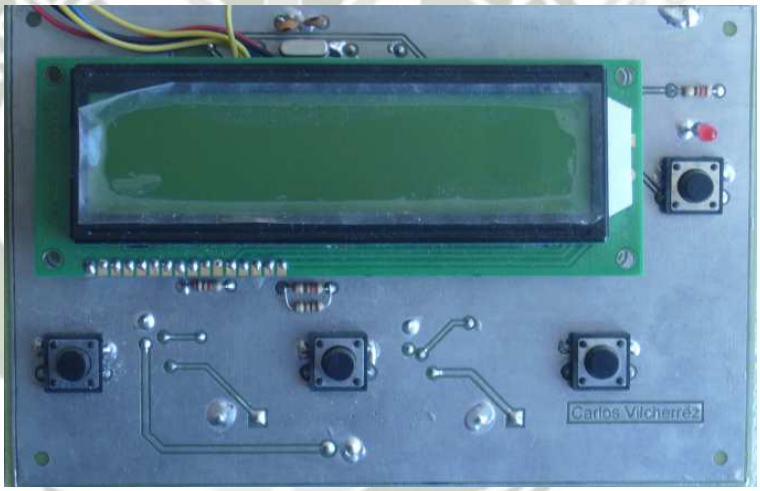


FIGURA 7.21. Placa de Mando Cara Superior

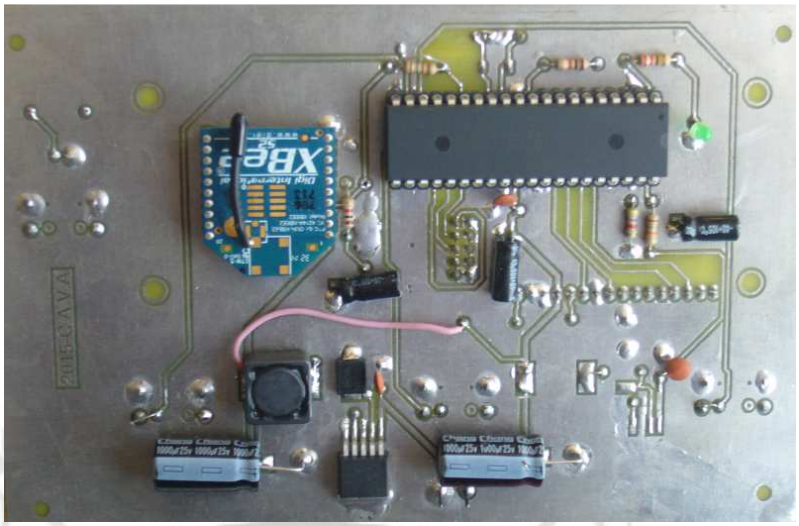


FIGURA 7.22. Placa de Mando Cara B

a) Circuito 1: Regulador de Voltaje

En el funcionamiento de este circuito usaremos dos reguladores de voltaje uno para el adaptador de ultrasónicos de 5v. y otro para el microcontrolador de 3.3V, necesario para los módulos xbee, los que veremos a continuación:

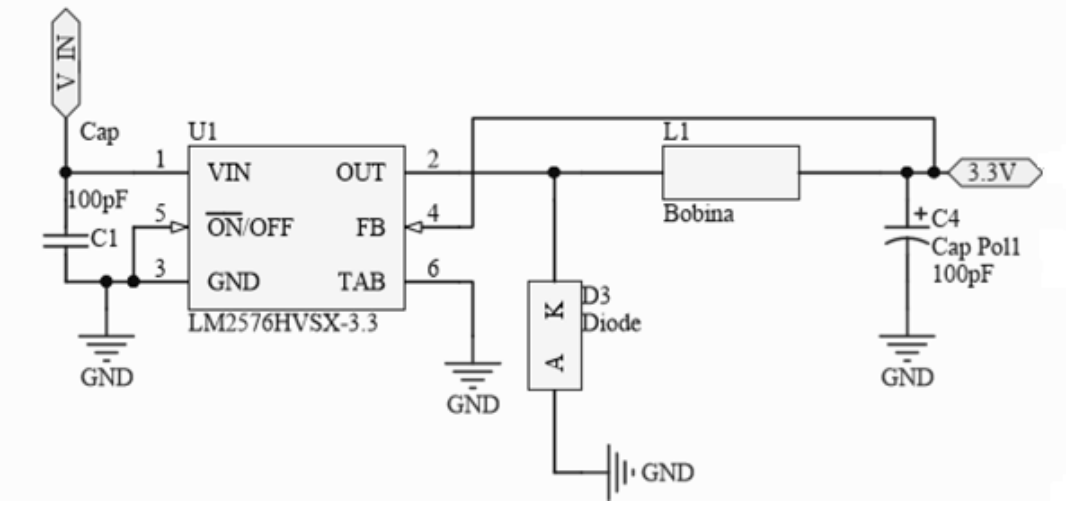


FIGURA 7.23. Esquemático del circuito reguladore de voltaje.

De igual manera que en la placa maestra, a continuación apreciamos el regulador de voltaje armado en el circuito de control de mando:

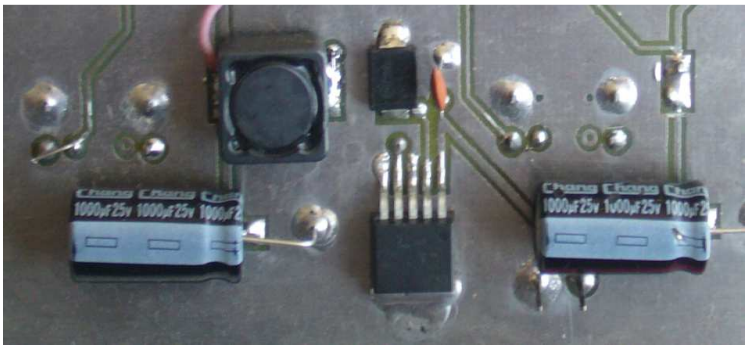


Figura 7.24. Circuito en placa de divisor de voltaje en el control de mando

b) Microcontrolador:

Básicamente el funcionamiento de este microcontrolador está explicado y desarrollado en el apartado anterior en el microcontrolador de la placa maestra, sin embargo lo más importante en esta placa, es que al recibir los datos desde la placa de planta, este tiene que tomar decisiones y esto se hace mediante los pulsadores, los cuales hacen la automatización de manera inalámbrica mediante radiofrecuencia, gracias a un monitoreo previamente hecho en la pantalla LCD.

A continuación explicaremos cómo y cuáles fueron los principios para el trabajo eficiente de los pulsadores.

Rebotes: Los pulsadores son interruptores mecánicos imperfectos, en el cierre de contactos se producen micro-rebotes y la señal que recibe el microcontrolador oscila entre 0v y 5v por un intervalo de 20 a 40

milisegundos aproximadamente. Este comportamiento provoca en nuestros programas una pulsación múltiple no deseada (la mayoría de las veces). Existen diferentes técnicas para parchar este problema, veremos la más sencilla y es la de preguntar si el botón fue pulsado, hacer una espera de 20ms y luego preguntar si el botón sigue pulsado.

Cada 20ms hay una interrupción (no existe rebote del pulsador en el lenguaje de programación), entonces nos manda a la interrupción y pregunta:

¿Qué se ha presionado?, para saber qué acción se va a tomar, si no hay nada nuevo regresa a lo que se estaba realizando anteriormente, para esto existe un flag de aviso.

Primero los pulsadores trabajan con un XOR y luego con un AND, de esta manera se asegura que se tome la acción establecida, lo coloca en un paso anterior si hay algún cambio, elimina los rebotes, para tomar una acción que necesita obligatoriamente un pulso nuevo.

Además este microcontrolador cuenta con una opción que veremos en la pantalla LCD de la actualización de la batería cada cierto tiempo, para ello se utiliza el otro Timer/counter al actualizar la pantalla cada medio segundo. Esta visualización de la batería es muy importante dado que el control de mando no puede apagarse ya que todo el proceso dejaría de ser controlado.

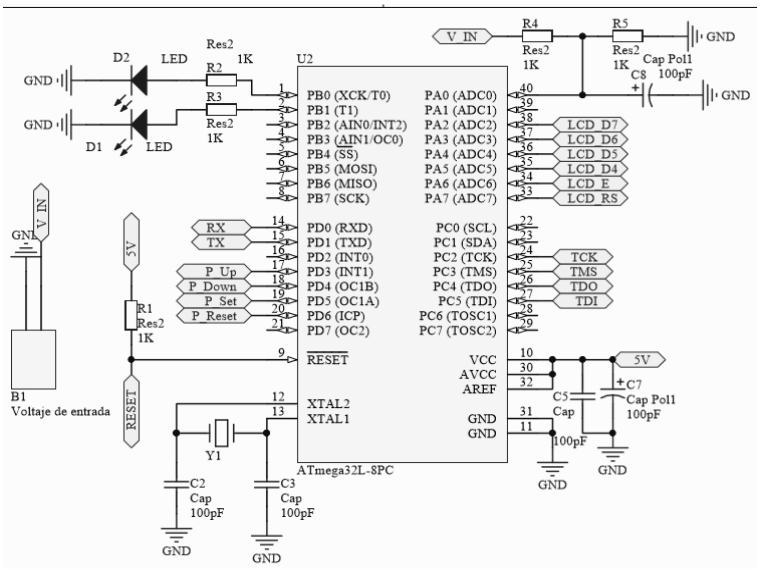


FIGURA 7.25. Esquemático del circuito del microcontrolador 16A

Ahora podemos apreciar el circuito del microcontrolador armado en el mando:

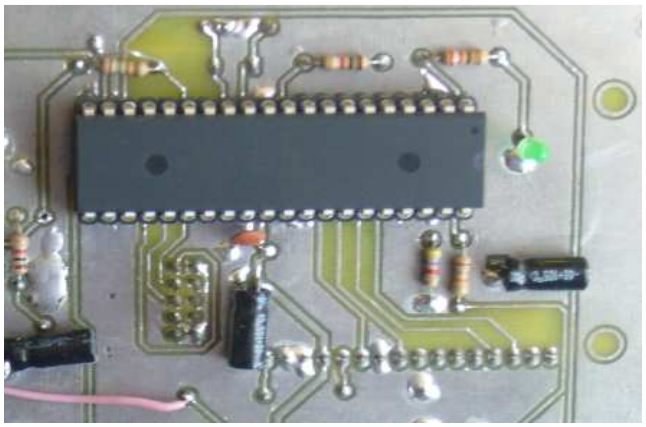


FIGURA 7.26. Circuito en placa del microcontrolador 1284P

CAPITULO VIII: PRUEBAS DE FUNCIONAMIENTO

8.1. PRUEBAS EN EL PID PARA LA CALIBRACIÓN

Lo primero que tuvo que probarse en este proyecto fue el funcionamiento correcto del PID en el calefactor, para ello se hizo unas pruebas para poder calibrar dicho PID las cuales presentaremos a continuación en forma de tablas, pero primeramente mostraremos como fue el comportamiento del PID:

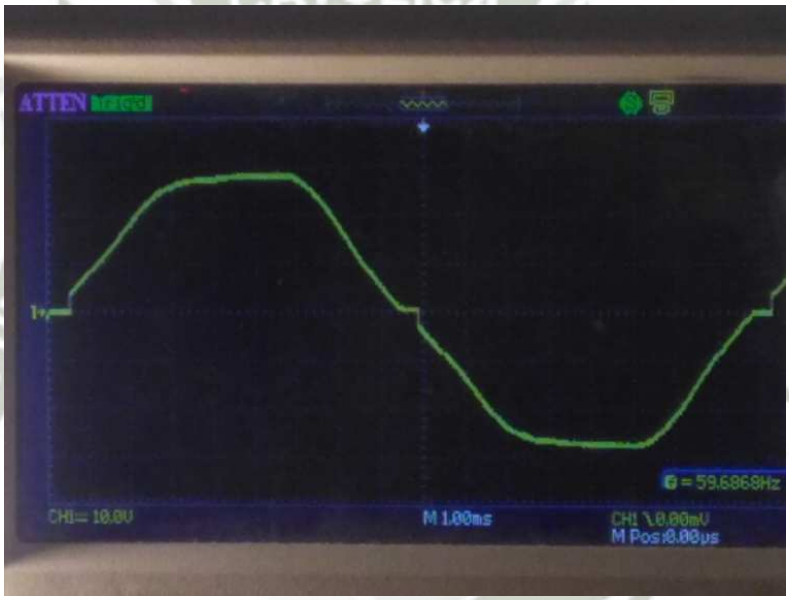


FIGURA 8.1. Prueba del funcionamiento del PID

Prueba del PID en el calefactor para estabilizar a 150°c:(TABLA 8.1.)

Primera Prueba		
	Tiempo (‘min,”seg)	Temperatura (°centrigrados)
Primera estabilización	3’ 15”	173°
	6’ 46”	146°
	8’ 30”	150°
Segunda estabilización	9’ 40”	153°
	11’ 30”	148°
	12’ 35”	150°
Tercera estabilización	13’ 30”	152°
	15’ 40”	148°
	17’ 10”	150°

Prueba del PID en el calefactor para estabilizar a 160°c: (TABLA 8.2.)

Segunda Prueba		
	Tiempo (‘min,”seg)	Temperatura (°centrigrados)
Primera estabilización	2’ 55”	173°
	5’ 30”	156°
	7’ 10”	160°
Segunda estabilización	8’ 00”	164°
	10’ 20”	157°
	11’ 30”	160°
Tercera estabilización	12’ 23”	162°
	14’ 20”	158°
	15’ 45”	160°

Prueba del PID en el calefactor para estabilizar a 170°c: (TABLA 8.3.)

Tercera Prueba		
	Tiempo (‘min,”seg)	Temperatura (°centrigrados)
Primera estabilización	2’ 43”	179°
	5’ 10”	166°
	6’ 22”	170°
Segunda estabilización	7’ 05”	174°
	9’ 40”	166°
	10’ 30”	170°
Tercera estabilización	12’ 40”	173°
	14’ 35”	167°
	15’ 50”	170°

Lo que apreciamos en las anteriores tablas es el tiempo que transcurre mientras va variando la temperatura y el PID intenta estabilizar el calefactor por eso en cada estabilización se puede observar que la primera medición es una temperatura máxima a la que llega el calefactor donde el PID trata de actuar para que no tenga un pico muy alto, y la segunda medición es la temperatura más baja que logra tomar el calefactor que de igual manera el PID trata de que la caída sea lo menos abismal posible; entonces cada vez que se fue haciendo estás mediciones se iban variando las constantes del PID, hasta finalmente cambiarlas por las que se calcularon al comienzo del capítulo de diseño.

A continuación se mostrará la prueba final donde el PID ya está con las constantes calculadas a una temperatura óptima para el proceso

Prueba General del PID para 150°c: (TABLA 8.4.)

Segunda Prueba		
Primera estabilización	Tiempo (‘min,”seg)	Temperatura (°centrigrados)
	4’ 15”	187°
	8’ 33”	144°
	9’ 30”	150°
Segunda estabilización	10’ 30”	156°
	12’ 50”	147°
	14’ 21”	150°
Tercera estabilización	15’ 22”	154°
	17’ 35”	148°
	19’ 12”	150°

En esta última tabla, podemos ver el control del PID, al variar la temperatura se da de forma un poco más lenta a las anteriores pero más estable y no con un cambio tan rápido, sin embargo es la que al momento de estabilizarse, esta se mantiene en su temperatura a diferencia de las otras que tienen un tiempo más rápido de cambio pero aún estables siguen variando mínimamente.

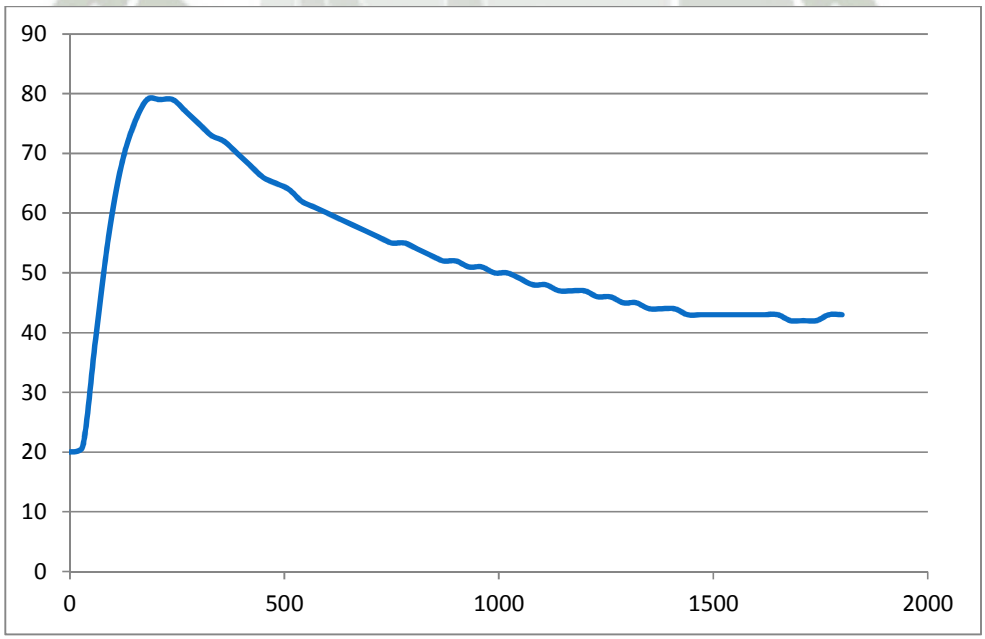
8.2. PRUEBAS DEL PID UNA VEZ CALIBRADO A DIFERENTES TEMPERATURAS

A continuación tendremos la tabla que al igual que las anteriores tablas para la calibración del PID están en función del tiempo y la temperatura, sin embargo en estas pruebas lo que se quiere ver ese funcionamiento a diferente temperaturas del PID una vez ya calibrado, se hicieron tres pruebas las cuales son las siguientes:

Prueba del PID calibrado a 40c°: (TABLA 8.5)

Tiempo (s)	Temperatura (c°)
0	20
30	21
60	39
90	56
120	68
150	75
180	79
210	79
240	79
270	77
300	75
330	73
360	72
390	70
420	68
450	66
480	65
510	64
540	62
570	61
600	60
630	59
660	58
690	57
720	56
750	55
780	55
810	54
840	53
870	52
900	52
930	51
960	51
990	50
1020	50
1050	49
1080	48
1110	48
1140	47
1170	47

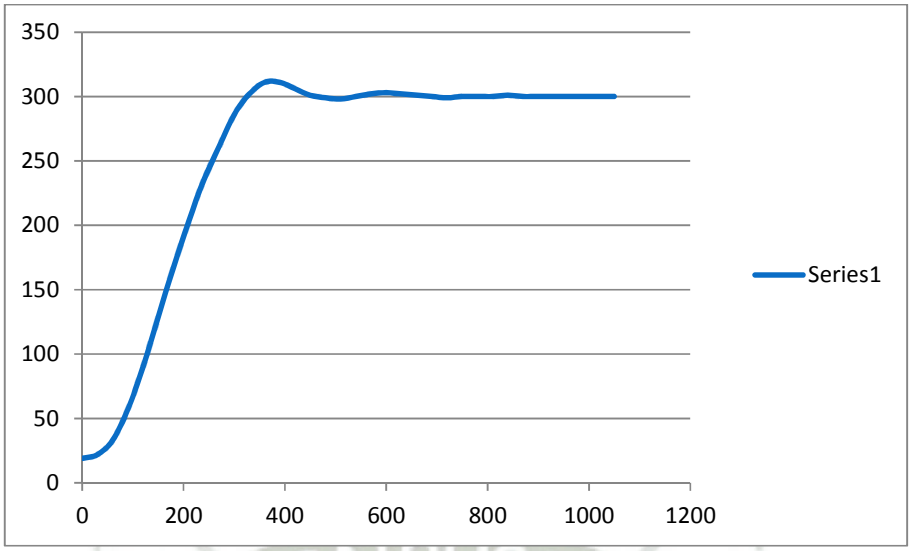
Tiempo (s)	Temperatura (c°)
1200	47
1230	46
1260	46
1290	45
1320	45
1350	44
1380	44
1410	44
1440	43
1470	43
1500	43
1530	43
1560	43
1590	43
1620	43
1650	43
1680	42
1710	42
1740	42
1770	43
1800	43



GRÁFICA 8.1. prueba con PID calibra a 40° c

Pruebas del PID calibrado a 300°c: (TABLA 8.6.)

Tiempo (s)	Temperatura (c°)
0	19
30	22
60	33
90	57
120	90
150	129
180	167
210	203
240	235
270	261
300	286
330	302
360	311
390	311
420	306
450	301
480	299
510	298
540	300
570	302
600	303
630	302
660	301
690	300
720	299
750	300
780	300
810	300
840	301
870	300
900	300
930	300
960	300
990	300
1020	300
1050	300



GRÁFICA 8.2. prueba con PID calibra a 300° c

Anteriormente lo que hemos probado es variar la temperatura para que pueda estabilizarse en diferentes grados centígrados en este caso en 40°c y 300°c.

8.3. PRUEBA DEL PROCESO AUTOMATIZADO POR RADIOFRECUENCIA GENERAL

El primer punto para probar correctamente todo el proceso será la conexión entre el módulo (placa general) y el control de mando, esto quiere decir que primeramente debe existir la conexión RF entre las dos placas para poder dar las órdenes y automatizar el proceso, caso contrario no existirá ninguna automatización.

FIGURA 8.2. Prueba de conexión de módulos RF

Una de las pruebas más sencillas para saber si los datos de la placa principal están llegando al control de mando es la que se puede apreciar en la imagen anterior, cuando no existe conexión la pantalla LCD se encuentra sin datos es por eso que no se aprecia ninguna medición en ninguna de las dos primeras opciones observada, sin embargo al encontrar comunicación con el módulo XBEE este empieza a recibir datos, es por ello que se puede visualizar en la imagen de abajo la medición tanto de temperatura como de llenado de Tolva A.

Seguidamente se encenderá el proceso y se le añadirá pequeños trozos de plástico de botellas de aproximadamente 1.5cm de largo por 0.5cm de ancho, los cuales fueron cortados anteriormente, simulando un previo proceso de triturado del plástico, como se puede ver a continuación.

FIGURA 8.3. Material de plástico picado para las pruebas correspondientes

Una vez iniciado el proceso ya con el material dentro comienza a funcionar toda la automatización y el monitoreo, comenzando por el sensor PIR de movimiento el cual tiene un tiempo de aproximadamente 60 segundos de espera, si en ese tiempo transcurrido no entra ningún material entonces todo el proceso será apagado.

FIGURA 8.4. Ingreso de material al proceso

Seguidamente podemos apreciar el primer sensor ultrasónico en la pantalla LCD del control de mando, el cual va midiendo la altura de llenado de la tolva A como podemos apreciar en la imagen siguiente, esto para prevenir un desbordamiento de la tolva en caso de cualquier falla en el proceso.

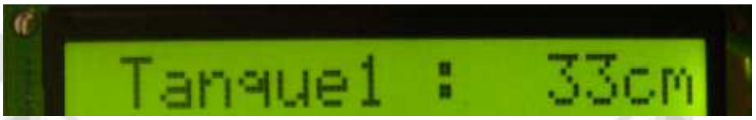
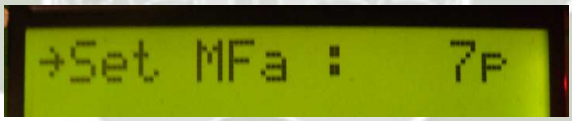
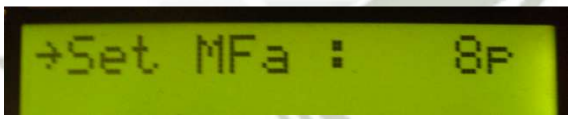


FIGURA 8.5. Medición del llenado de tolva principal

A continuación se muestra en el control de mando la variación del motor de la faja, cabe resaltar que en este informe solo se puede apreciar la variación en la pantalla LCD del control de mando, por motivos obvios, sin embargo la prueba se hará al momento de exponer dicho proyecto.





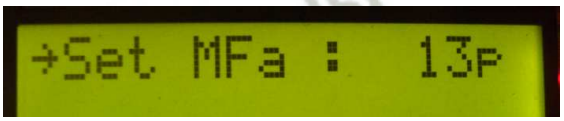


FIGURA 8.6. Variación del motor por PWM mediante pulsos

La siguiente parte del proceso que se muestra será el segundo sensor ultrasónico, el cual de la misma manera que el primero, este monitorea y sensa el llenado de la tolva B, como se dijo anteriormente en el caso que exista alguna avería en el proceso. Podemos apreciar la altura de llenado en la siguiente imagen:

FIGURA 8.7. Medición de llenado de tolva secundaria

Llegamos al extrusor, el cual funciona de la siguiente manera, el calefactor es independiente del motor, o sea una vez encendido este, no quiere decir que el motor del espiral del extrusor también comience a trabajar; una vez encendido el proceso, el calefactor también enciende pero a la temperatura elegida por el control de mando y es ahí donde actúa el PID.

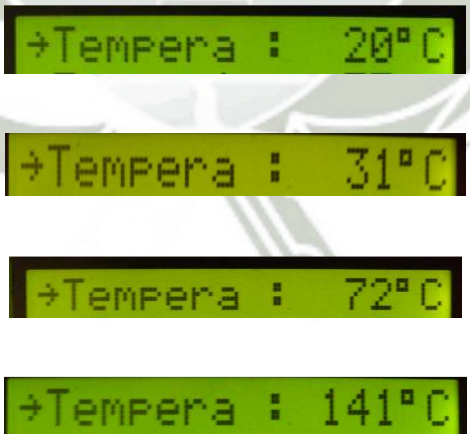


FIGURA 8.8. Variación de temperatura en la pantalla LCD

Finalmente, como se dijo el motor del espiral del extrusor es independiente del calefactor entonces cada vez que ingrese una cantidad adecuada de material al extrusor, el control de mando envía una señal para que el motor comience a funcionar, esto se da debido a que el extrusor al ser el último trabajo del proceso, en el caso de que funcione continuamente corre el riesgo de que en un momento no ingrese material al extrusor entonces que ese motor funcione mientras ocurre eso es una pérdida de consumo eléctrico y tiempo de vida de la máquina, también si existe alguna avería antes del extrusor ya sea en la tolva B, en la faja o en la tolva A, pues de igual manera esta parte del proceso seguirá funcionando sin razón alguna; además de ser el final del proceso, y en este caso el proceso deja el plástico como materia prima lista para ser usada de cualquier forma posible y es casi un hecho que después de este extrusor exista un último proceso de moldeo donde el extrusor no puede trabajar de forma continua ya que casi todos los procesos de moldeo necesitan una pausa para poder cambiar el molde o la forma siguiente que se está requiriendo ya sean baldes, tachos, tapas, etc.

Este punto del calefactor se puede apreciar en el control de mando como varia la temperatura; sin embargo al igual que en la faja, la activación del motor del extrusor se puede ver en el control de mando más no el movimiento en sí, sin embargo al momento de la presentación del proyecto este será probado.

FIGURA 8.9. Elección de la temperatura a la que se quiere trabajar

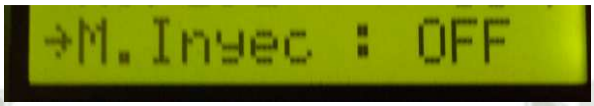


FIGURA 8.10. Activación de motor del extrusor

CAPITULO IX: RESULTADOS DE LA INVESTIGACIÓN

(1). Primeramente se pudo monitorear y controlar las variables requeridas para el proceso, dichas variables podemos apreciarlas en el siguiente cuadoro

VARIABLES	PUNTOS DE AUTOMATIZACION	ETAPA DEL PROCESO
a) Monitoreadas		
Movimiento	Tolva A	Inicio del proceso
Altura de material en la tolva	Tolva A	Inicio del proceso
	Tolva B	extrusión
b) Controladas		
Velocidad de motores	Motor de la faja de transporte	Transporte
	Motor de activación del extrusor	Extrusión
Temperatura del extrusor	Calefactor del extrusor	Extrusión

TABLA 9.1. Variables monitoreadas y controladas en el proceso de reciclaje de plásticos.

a) Resultados en variables monitoreadas

- **Movimiento**: Se logró controlar de manera óptima el inicio del proceso gracias al sensor de movimiento PIR, el cual funciona cambiando el nivel lógico de un pin de la siguiente manera:
  - ✓ Sensor en reposo (no hay movimiento) → nivel bajo (0)
  - ✓ Sensor activo (detecta movimiento) → nivel alto (1)

- **Altura de llenado en la tolva**: Se logró controlar de manera óptima, eficaz y eficiente el llenado de material dentro de la tolva, de acuerdo a la variable de manera siguiente:

- ✓ Altura entre 29cm – 5cm de alto.
- ✓ Al existir menos de 5cm de altura entre el sensor y el material de llenado este alerta al proceso

b) Resultados de variables controladas

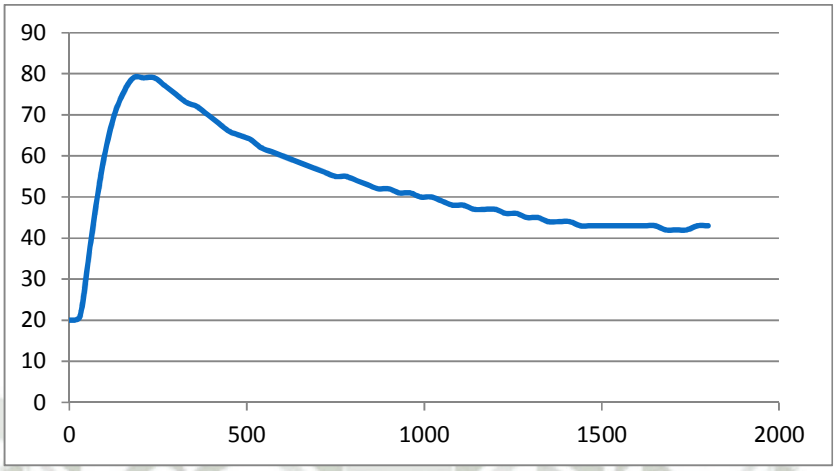
- **Velocidad en los motores**: La velocidad en los motores pudo ser variada y controlada óptimamente, de acuerdo a la variable siguiente de velocidad:

  - ✓ Si bien la velocidad en motores se mide en RPMs, en nuestro caso se dio un rango de números.
  - ✓ La velocidad de motores se da entre velocidades por pulso de 7 a 14

- **Temperatura del extrusor:** La temperatura pudo ser variada y controlada óptima, **eficaz** y lo más eficientemente posible más por ser un punto en el que la seguridad tiene que ser prioridad, y se hizo de la siguiente manera:

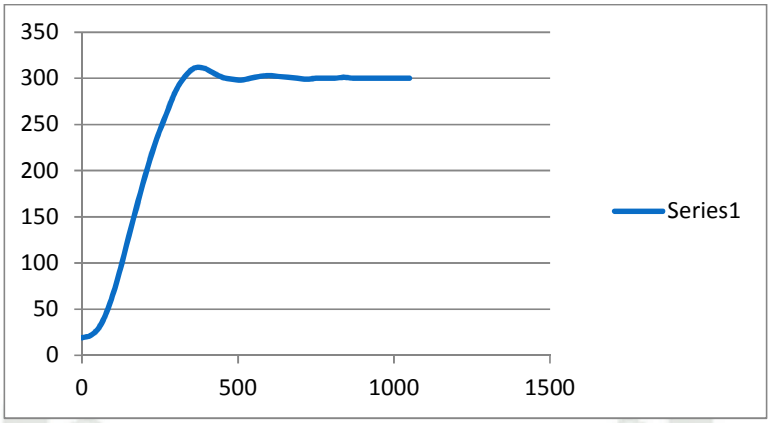
  - ✓ La temperatura se varió en grados centígrados desde 19°c hasta 300°c
  - ✓ De acuerdo a las pruebas anteriores al variar la temperatura en este primer caso a 40°c, el control PID tarda mucho más tiempo en controlar la temperatura baja, esto es por inercia debido a que al encender el calefactor este tiende a calentarse de manera rápida sin embargo el PID empieza a actuar y no deja que ascienda bruscamente la temperatura, es por este motivo que demora un poco

más de tiempo en estabilizar en este caso 1800 segundos (30 minutos).



GRÁFICA 9.1.Prueba PID a 40°c

- ✓ En la siguiente prueba a una temperatura variada de 300°c, obtuvimos que en este caso el control PID actúa de una manera más rápida, ya que al ser la temperatura más alta el PID tiene un poco más de tiempo para poder actuar y estabilizar la temperatura elegida, y en este caso el control total dela temperatura a 300°c se da en aproximadamente 900 segundos (15 minutos), como podemos apreciar en la gráfica obtenida:



GRÁFICA 9.2.Prueba PID a 300°c

(2). El programa desarrollado logró funcionar de manera óptima a los requerimientos dados por el proceso automatizado, el cual se pudo apreciar con las pruebas realizadas.

(3). La propuesta de utilizar comunicación por radiofrecuencia fue eficaz para el proceso dado, ya que no se tuvo inconvenientes de alcance ni de potencia de emisión entre los módulos.

(4). Como resultado de esta investigación se han identificado hasta cuatro efectos positivos de la automatización en la industria de reciclado de plásticos, cuyas aplicaciones generan como efecto mayor la disminución de costo en la industria y el incremento de los beneficios económicos. Estos efectos positivos son:

- Menor consumo de energía
- Mayor vida útil de maquinaria
- Menor requerimiento de mano de obra
- Mayor seguridad industrial
- Menor requerimiento de mantenimiento

a) Menor consumo de energía

En una planta industrial es de suma importancia el consumo de energía de las máquinas, entonces si estas pueden usarse solamente en el tiempo de trabajo y no dejarlas funcionando cuando no sea necesario entonces se ahorra el gasto en el consumo de energía de más.

En nuestro caso, el menor consumo de energía se produce por el eficiente control de la velocidad de los motores en las diversa fases del proceso productivo:

b) Mayor vida útil de la maquinaria

Cuando las máquinas siguen trabajando aunque no exista ningún material para ser procesado el desgaste es más rápido. Con la automatización se evita este desgate y, por lo tanto, aumenta la vida útil de las maquinarias.

c) Menor requerimiento de mano de obra

En nuestro caso estos menores requerimientos de mano de obra por efectos de la automatización se darían en los procesos de encendido y apagado de planta, así como en los momentos que se generen desperfectos y sean necesarias las labores de mantenimiento.

d) Mayor seguridad industrial

En el ambiente industrial, siempre hay un margen de riesgo cuando un técnico entra a planta (específicamente en las maquinarias); este riesgo

se incrementa cuando se generan desperfectos en el funcionamiento de las maquinarias.

En nuestro caso los mayores riesgos se generarían por deficiencias de funcionamiento del extrusor. Con la automatización, se reduce casi totalmente el riesgo de un accidente o incidente en la planta, debido a que al ser la temperatura cambiada automáticamente desde el control de mando, no hay necesidad de que algún técnico u obrero ingrese al extrusor cuando éste se encuentre en funcionamiento.

e) Menor requerimiento de mantenimiento

La automatización reduce o evita fallas en las diversas fases del proceso productivo, por lo tanto las labores de mantenimiento se reducen sustantivamente y se aumenta la eficiencia del funcionamiento de la planta en general.

(5). Finalmente tendremos el producto final que se obtuvo con este proceso en función a las velocidades de motores, temperatura del extrusor y tiempos de trabajo; las características de esta materia prima como resultado son: color medio gris (debido a la suciedad dentro del extrusor), bastante liviano, amorfo (por no haber molde después del extrusor). Como podemos apreciar a continuación:

FIGURA 9.1. Producto final en el proceso automatizado

CAPITULO X: COMPARACIÓN CON OTROS PROCESOS

10.1. COMPARACION CON PROCESO MECÁNICO

Con esto estamos diferenciando dos tipos de trabajo en el mismo proceso, y lo que se quiere demostrar es la mejora del proceso automatizado por radiofrecuencia en comparación al proceso mecánico tanto en eficiencia, eficacia y optimización.

Este proceso de reciclaje de plástico ha sido separado en dos fases:

Fase 1:

- Ingreso de Material a la planta.
- Tolva Principal.

Fase 2:

- Faja transportadora
- Tolva pequeña (tolva secundaria)
- Extrusor.
- Producto Final.

Las mejoras del proceso automatizado por radiofrecuencia en relación al proceso mecánico se demuestran en 4 puntos claves, y son los siguientes:

(1) Ingreso de Material a la Planta (Tolva Principal):

Control de Movimiento

Proceso mecánico: En el inicio del proceso el material ingresará por una tolva grande en este caso manualmente, ya que en este módulo de planta obviamos el proceso de prensado y triturado. En este proceso mecánico la planta funcionará continuamente así ingrese o no material a la planta, es decir, así no haya material por procesar la planta seguirá trabajando.

Proceso Automatizado por radiofrecuencia: En el inicio del proceso donde ingresa el material, se tiene instalado un sensor PIR de movimiento el cual es un sensor piroeléctrico (pasivo) infrarrojo con un rango de hasta 5 metros de detección, un ángulo < 100°, salida activa alta a 3.3 V. y un voltaje de alimentación de 4.5 a 20 Vdc. Debido a la información que este sensor nos brinda, el proceso está configurado para apagarse después de un determinado tiempo; 60s en este caso.

El sensor de movimiento toma los datos, estos llegan al microcontrolador anteriormente ya configurado con el programa; los datos son enviados al transmisor (módulo xbee s2) para que éste los envíe por el medio de comunicación, en este caso la radiofrecuencia, en la banda de frecuencia de 2.4GHz estos datos son recibidos por el receptor (módulo xbee s2) y posteriormente se muestra la

información de falta de material, la cual es enviada a la pantalla del control de mando.

Control de Llenado

Proceso Mecánico: En el inicio del proceso el material ingresará por la misma tolva principal, de igual manera manualmente, donde el mayor problema ocurriría si se presenta algún desperfecto en algún proceso de la fase 1, ya sea en la trituradora, faja transportadora o tolva B. El volumen del llenado de la tolva A desbordaría ya que no hay forma de que alguna parte del proceso se detenga, incluso si deja de ser llenada la tolva A, por no enviar información de la tolva o por no estar automatizada.

Proceso Automatizado por Radiofrecuencia: Al automatizar el proceso por radiofrecuencia, al colocar el material que ingresa en la tolva A, el volumen de llenado de esta tolva se controla mediante un sensor ultrasónico HC-SR04, ya dadas sus características anteriormente, el cual nos mide la altura del volumen de llenado de la tolva principal. Gracias a la información dada por este sensor, en caso ocurriera algún desperfecto previo o posterior a este punto del proceso, nos envía una señal al control de mando indicando cuando la altura del volumen de la tolva se encuentre a 5 cm del sensor; entonces, quien este encargado de ese proceso sabrá que está por llenarse esa tolva y dará un mensaje a quien sea el encargado de

proveer el material a la tolva para que en ese momento deje de hacerlo. La fase 2 no es deshabilitada porque no siempre puede existir desperfecto en la fase siguiente, si no que en un momento determinado puede que se haya acumulado demasiado material y que no se haya podido procesar todo óptimamente, para lo cual es factible que el técnico encargado del proceso sepa de ello para que pueda mandar un mensaje y se deje de alimentar la tolva A, pero el siguiente proceso debe seguir trabajando para descongestionar la tolva A. Por este motivo no se para el proceso en general.

(2) Transporte de material picado (Faja Transportadora)

Proceso Mecánico: Para que el material picado llegue hasta la tolva secundaria, se necesita un medio de transporte; lo más conveniente en estos casos es un faja transportadora. Esta faja transportadora, en nuestro modelo, funciona como anteriormente ya lo mencionamos con un motor de hasta 24Vdc a 1 amperio. En un modelo industrial esta faja tendría que estar aislada ya que al ser triturado el plástico éste puede botar partículas pequeñas del plástico triturado, aunque es poco o nulo el porcentaje de esas partículas, nunca deja de ser importante prevenir; esta faja transportadora en un modelo mecánico funcionaría a una velocidad constante y su motor a su potencia máxima.

Proceso Automatizado por Radiofrecuencia: La velocidad de la faja transportadora sería variable y la potencia de su motor no sería

máxima. La velocidad de la faja sería variada mediante el control de mando, y esta variación se da por PWM, se hace variable la velocidad, debido a que el proceso dependiendo del material, ambiente o cualquier otra cosa, pueda necesitar de una potencia en la faja baja, media o tal vez alta, y al poder ser variada desde un control inalámbrico el trabajo se hace más sencillo y eficiente.

La faja transportadora debe trabajar también en función de la tolva secundaria ya que, mientras la altura medida del volumen de llenado sea mayor, el operador del control de mando varía la velocidad de la faja transportadora, de acuerdo a la información en centímetros de la altura de llenado de esta tolva, y esta variación de velocidad en la faja sería inversamente proporcional al llenado de la tolva; es decir, a más altura tenga el volumen de llenado de la tolva menor será la velocidad de la faja.

(3) Ingreso de Material Picado (Tolva Secundaria)

Proceso Mecánico: El material picado llega hasta la tolva B mediante la faja transportadora, en un proceso mecánico, de haber algún desperfecto en el extrusor, el volumen del llenado de la tolva B desbordaría.

Proceso Automatizado por Radiofrecuencia: El volumen de llenado de esta tolva se controla mediante un sensor ultrasónico HC-SR04, anteriormente ya mencionado, al igual que sus características, el cual nos mide la altura del volumen de llenado de

la tolva secundaria. Este sensor funciona según el tiempo que transcurre entre el envío y la recepción del ultrasonido. Gracias a la información dada por este sensor, en caso ocurriera algún desperfecto en el extrusor, el proceso está programado para que nos envíe una señal al control de mando indicando cuando la altura del volumen de la tolva se encuentre a 5 cm del sensor. Entonces el técnico debe dejar de alimentar esta tolva, ya sea dejando de alimentar la tolva principal o variando la velocidad del motor de la faja a la velocidad a más lenta.

(4) Proceso de Extrusión:

Proceso Mecánico: En un proceso mecánico de extrusión, la máquina trabajaría a una velocidad constante, dada por el motor, y de igual manera trabajaría a una sola temperatura para la extrusión del plástico. En el caso se quiera variar la temperatura del extrusor, ya sea por cambio de material o por algún cambio externo de la temperatura, se tiene que hacer el cambio propuesto manualmente, y al trabajar con temperaturas tan altas mayores a 100° C hay un riesgo muy alto para la mano de obra que realice dicho trabajo.

Proceso Automatizado por Radiofrecuencia: Al ser automatizado por radiofrecuencia, el motor de la máquina sería accionado cada vez que ingrese el material suficiente dependiendo del molde o el proceso de moldeo posterior, el cual en este caso no pudo simularse, y en el caso de la temperatura, esta sería automatizada, o

sea uno puede elegir la temperatura a la que necesite trabajar, ya sea por un material diferente porque no todos los tipos de plásticos se licuan a la misma temperatura, o también como se mencionó en el proceso mecánico puede haber un cambio de temperatura externo. Por ello, el riesgo y el trabajo disminuyen, ya que el extrusor es monitoreado y automatizado por un control de mando en radiofrecuencia.

10.2. COMPARACIÓN CON PLC

Primeramente un PLC es como sus siglas lo dicen un programador lógico programable y son más conocidos por ser un sistema de control automático que trabaja bajo una secuencia almacenada en memoria de instrucciones lógicas; en otras palabra está diseñado para controlar procesos secuenciales ejecutados en un ambiente industrial, en otras palabras que van asociados a la maquinaria que desarrolla procesos de producción y controlan su trabajo.

Aunque estos son bastante viables y su desempeño es óptimo para el trabajo industrial, pues cuentan con algunas desventajas como:

- La inversión inicial es mayor que en el caso de los relés, aunque ello es relativo en función del proceso que se desea controlar. Dado que el PLC cubre de forma correcta un amplio espectro de necesidades, desde los sistemas lógicos cableados hasta el microprocesador, el diseñador debe conocer a fondo las prestaciones y limitaciones del PLC. Por tanto, aunque el coste inicial se debe tener en cuenta a la

hora de decidirnos por uno u otro sistema, conviene analizar todos los demás factores para asegurarnos una decisión acertada.

Sin embargo, la mayor desventaja de un PLC es sin duda alguna el costo tan elevado tanto del mismo PLC como de instalación, es por ello que el proyecto presente es una nueva opción para el control automatizado en la industria. Primeramente por ser un diseño no tan complicado de implementar, con componentes que existen en nuestro medio y fáciles de encontrar, además como fue demostrado de ser una opción óptima, eficaz y eficiente para la automatización.

Finalmente existen algunos otros tipos de tecnologías o sistemas de control automático sin embargo estos no son tan completos como el PLC, y son comúnmente usados en pequeñas industrias para procesos simples y cortos, es por eso que la comparación principal se centra en el proyecto planteado con el PLC.

CAPITULO XI: CONCLUSIONES

(1). Se logró el objetivo de diseñar e implementar un módulo para un proceso automatizado de reciclaje de plástico por radiofrecuencia, teniendo una funcionalidad óptima y eficaz.

(2). Tras tener problemas en el control de motores debido a los trasientes que estos presentaban, se rediseño un modelo de placa con mayor robustez, seguridad y eficiencia para el proceso planteado.

(3). Los sensores elegidos para este proceso funcionaron de forma óptima en todos los sentidos, tanto el de movimiento, ultrasónicos y de temperatura, las características de estos fueron más que suficientes para el correcto funcionamiento del proceso.

(4). El control en el PID tiene un funcionamiento proporcional en función a la eficiencia de este, debido a la inercia del calor al encender el calefactor, a menor temperatura la eficiencia baja a un margen de error de 1° o 2° grados centígrados, y a mayor temperatura la eficiencia de control mejora hasta un punto exacto, cabe resaltar que no se hizo pruebas registradas a más de 300 grados por motivos de seguridad aunque en pruebas sin registrar anteriores si logro controlarse hasta 500°c.

(5). El proceso presentado junto con la automatización, lograron obtener un producto final, el cual es totalmente reciclable, y dicho proceso podría presentar para implementarlo de manera industrial.

(6). En el ámbito de seguridad industrial, se logró diseñar un sistema de automatización totalmente seguro, ya que al ser la comunicación por radiofrecuencia el riesgo de accidente o incidente disminuye considerablemente.

(7). La comunicación por radiofrecuencia con los módulos XBEE pudo darse hasta los 100 metros, esto debido alcance de dichos módulos, sin embargo la capacidad de esta comunicación es ilimitada dependiendo netamente del transmisor y receptor usados.

CAPITULO XII: MEJORAS O RECOMENDACIONES

(1). Si se quiere implementar este proceso a escala real en un modelo industrial, es de suma importancia primeramente regirse por las normas y estatutos dados, ya que por tratarse de un módulo de simulación a escala pequeña esta no cuenta con tales normas y reglas.

(2). Para un mejor proceso de manera más completa, se puede añadir un proceso de moldeo al final después del proceso de extrusión, lo que en este caso no se hizo por motivos económicos.

(3). Una mejora bastante interesante sería el de ingresar todo el monitoreo y control del sistema a la red de internet, lo cual con un poco más de investigación es posible, ya que estos módulos XBEE tienen mucha variedad y formas de trabajo.

(4). Puede automatizarse el proceso de la forma más convencional y conocida, sin embargo en este tipo de automatización no se podría variar parámetros, sería factible una complementación entre el diseño propuesto y el ya existente.

CAPÍTULO XIII: REFERENCIAS BIBLIOGRÁFICAS

(1). Benjamin C. Kuo: Sistemas de Control Automático, séptima edición.

(2). Virginia Mazzone: Controladores PID, Universidad Nacional de Quilmes.

(3).Manuel Sierra Perez: Electrónica de Comunicaciones (1994)

(4). Cerrón Salcedo, Juan Diego Ernesto y Moreno Arévalo, Edmundo Oswaldo: Tesis " Automatización y diseño del sistema de la planta de balanceados en los proceso de molienda, mezclado y peletizado de la empresa de alimentos procesados S.A. e implementación del prototipo del módulo de acondicionamiento para el proceso de peletizado".

(5). Bach. Johan Edwin Vela Moscoso: Sistema de control y supervisión de domótica basada en una estación remota con modem GSM".

Páginas Visitadas

- http://www.raco.cat/index.php/RevistaMetodosNumericos/article/viewFile/68700/101434
- http://www.edutecne.utn.edu.ar/microcontrol_congr/industria/mtodob~1.pdf
- http://qubits.wordpress.com/2009/04/04/esquematico-de-conexionado-y-montaje-de-modulos-xbee/
- http://voltiosybytes.blogspot.pe/2014/02/control-de-potencia-en-ac-con-triac-y.html
- http://es.slideshare.net/yagamyxD/informecontrol-de-potencia-por-angulo-de-disparro
- http://microcontroladores-mrelberni.com/pwm-timer1-avr-modo-rapido/

- http://ocw.usal.es/eduCommons/ensenanzas-tecnicas/materiales-ii/contenidos/PLASTICOS.pdf

- https://sites.google.com/site/picuino/ziegler-nichols

ANEXOS

ANEXO 1.- COMPORTAMIENTO DEL CALEFACTOR

Tabla completa del comportamiento del calefactor en el extrusor

Tiempo (s)	0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300
Temperatura (°c)	29 29 29 30 31 33 37 41 47 53 62 71 81 92 104 117 130 143 157 172 183 195 207 219 230 241 251 260 269 278 287

Tiempo (s)	310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500 510 520 530 540 550 560 570 580
Temperatura (°c)	295 302 311 319 326 334 341 347 354 360 366 373 378 384 390 394 399 404 409 413 418 423 427 431 436 440 443 447

Tiempo (s)	590 600 610 620 630 640 650 660 670 680 690 700 710 720 730 740 750 760 770 780 790 800 810 820 830 840 850 860
Temperatura (°c)	451 455 459 464 468 472 476 480 484 487 491 493 496 500 503 506 508 512 514 516 519 521 524 526 529 532 534 536

Tiempo (s)	870 880 890 900 910 920 930 940 950 960 970 980 990 1000 1010 1020 1030 1040 1050 1060 1070 1080 1090 1100 1110
Temperatura (°c)	539 541 543 545 548 549 551 553 555 558 561 563 565 568 570 572 574 575 577 578 580 582 584 586 587

Tiempo (s)	1120 1130 1140 1150 1160 1170 1180 1190 1200 1210 1220 1230 1240 1250 1260 1270 1280 1290 1300 1310 1320 1330
Temperatura (°c)	589 590 592 594 595 597 598 600 600 602 604 606 608 609 611 612 614 616 617 619 620 622

Tiempo (s)	1340 1350 1360 1370 1380 1390 1400 1410 1420 1430 1440 1450 1460 1470 1480 1490 1500 1510 1520 1530 1540 1550
Temperatura (°c)	623 624 625 626 627 627 628 630 631 632 633 635 636 637 639 640 641 642 643 644 645 646

Tiempo (s)	1560 1570 1580 1590 1600 1610 1620 1630 1640 1650 1660 1670 1680 1690 1700 1710 1720 1730 1740 1750 1760 1770
Temperatura (°c)	647 648 650 651 652 653 654 655 656 657 658 658 659 660 661 661 662 663 664 664 665 666

Tiempo (s)	1780 1790 1800 1810 1820 1830 1840 1850 1860 1870 1880 1890 1900 1910 1920 1930 1940 1950 1960 1970 1980 1990
Temperatura (°c)	667 667 667 667 668 668 668 669 670 670 671 671 672 672 672 672 672 672 672 672 671 671

Tiempo (s)	2000 2010 2020 2030 2040 2050 2060 2070 2080 2090 2100 2110 2120 2130 2140 2150 2160 2170 2180 2190 2200 2210
Temperatura (°c)	671 672 672 672 672 673 673 674 674 674 675 675 675 676 676 676 676 677 677 678 678 678

Tiempo (s)	2220 2230 2240 2250 2260 2270 2280 2290 2300 2310 2320 2330 2340 2350 2360 2370 2380 2390 2400 2410 2420 2430
Temperatura (°c)	679 679 679 679 679 680 680 680 680 680 680 680 681 681 681 681 681 681 681 681 680 680

Tiempo (s)	2440 2450 2460 2470 2480 2490 2500 2510 2520 2530 2540 2550 2560 2570 2580 2590 2600
Temperatura (°c)	680 680 680 680 680 680 680 680 680 681 681 681 681 681 681 681 681

ANEXO 2.- .- PROGRAMACIÓN HECHA EN C DEL PROCESO

PLACA MAESTRA

Recicladora PID

```
#define F_CPU 11059200UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "Control_PID.h"
#include "LCD2x16.h"
#include "Inicio_Micro.h"
#include "Modulos_Ultrasonicos.h"
#include "Usart.h"
unsigned char contador_ultrasonicos=0;
unsigned char contador_usart_tx=12;
ISR(TIMER2_OVF_vect)
{
	ptVariablesPID->tiempo_actual++;
	contador_ultrasonicos++;
	contador_usart_tx++;
	TCNT2=TIME_20_MS;
}
ISR(TIMER0_OVF_vect)
{
	contador_medio_grado++;
	if (contador_medio_grado>=medio_grado)
	{
		TRIAC_ON;
		contador_medio_grado=0;
		Detiene_Timer_Contador_Medio_Grado();
	}
	TCNT0=TIME_MEDIO_GRADO;
}
ISR(INT2_vect)
{
	Inicia_Timer_Contador_Medio_Grado();
}
int main(void)
{
	_delay_ms(50);
	Inicia_Puertos();
	TRIAC_OFF;
	LCD_Inicia();
	ptVariablesPID=&VariablesPID;
	ptMediciones=&Mediciones;
	ptConvertidor_Int_String=&Convertidor_Int_String;
	ptSensores_Ultrasonicos=&SensoresUltrasonicos;
	ptBuffer_Tx_Frames=&BufferTxFrames;
	ptBuffer_Rx_Frames=&BufferRxFrames;
	OFF_TRIGGER_TG;
	OFF_TRIGGER_TP;
	char pBuffer[100];
	medio_grado=358;
	ptVariablesPID->modo_trabajo=AUTOMATIC;
	ptVariablesPID->set_point=200;
	ptVariablesPID->tiempo_muestreo=50;
	Configura_Sintonizacion(ptVariablesPID,4.2,0.05,83.5);
```

```
	Configura_Limites_Salida(ptVariablesPID,0,100);
	USART_Init(71);
	Inicia_Interrupcion_Cruce_Cero();
	Configura_SPI_MAX31855();
	Inicia_Timer_Counter_20ms();
	Inicia_PWM_Motor();
	XBEE_RESET_OFF;
	Configura_Interrupcion_Ultrasonicos();
	sei();
    while(1)
    {
		Calcula_PID(ptVariablesPID);
		LCD_Goto_XY(8,1);
		PrintFloat((float)ptVariablesPID->salida_controlador,2,pBuffer);
		LCD_Write(pBuffer[0]);
		LCD_Write(pBuffer[1]);
		LCD_Write(pBuffer[2]);
		LCD_Write(pBuffer[3]);
		LCD_Write(pBuffer[4]);
		LCD_Write(pBuffer[5]);
	Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,(medio_grado/2))
;
		LCD_Goto_XY(0,2);
		LCD_Write(ptConvertidor_Int_String->centenas);
		LCD_Write(ptConvertidor_Int_String->decenas);
		LCD_Write(ptConvertidor_Int_String->unidades);
		if (contador_ultrasonicos>25)
		{
			contador_ultrasonicos=0;
		Control_Disparos_Sensores_Ultrasonicos(ptSensores_Ultrasonicos);
			LCD_Goto_XY(4,2);
			LCD_Write(ptMediciones->parametros[TANQUE1]);
			LCD_Write(ptMediciones->parametros[TANQUE1+1]);
			LCD_Write(ptMediciones->parametros[TANQUE1+2]);
			LCD_Write('c');
			LCD_Write('m');
			LCD_Write(' ');
			LCD_Write(ptMediciones->parametros[TANQUE2]);
			LCD_Write(ptMediciones->parametros[TANQUE2+1]);
			LCD_Write(ptMediciones->parametros[TANQUE2+2]);
			LCD_Write('c');
			LCD_Write('m');
		}
		Control_Planta(ptMediciones,ptVariablesPID);
		if (contador_usart_tx>25)
		{
			contador_usart_tx=0;
			Llena_Buffer_Tx(ptBuffer_Tx_Frames,ptMediciones);
			Inicia_Envio_Buffer_Tx(ptBuffer_Tx_Frames);
		}
		Verifica_Buffer_Rx(ptBuffer_Rx_Frames);
		Libera_Buffer_Rx(ptBuffer_Rx_Frames,ptMediciones);
    }
}
```

Control PID.c

```
#include "Control_PID.h"
const unsigned int angulos_potencia[101] PROGMEM={
	340,318,307,299,293,287,282,278,274,270,267,263,260,257,254,252,249,246,24
4,241,//20
	239,237,234,232,230,228,226,223,221,219,217,215,213,211,209,208,206,204,20
2,200,//40
	198,196,194,193,191,189,187,185,184,182,180,178,176,175,173,171,169,167,16
6,164,//60
	162,160,158,156,154,152,151,149,147,145,143,141,139,137,134,132,130,128,12
6,123,//80
	121,119,116,114,111,108,105,103,100, 96, 93, 90, 86, 82, 78, 73, 67,
61,53,41,19//101
};
void Inicia_Timer_Counter_20ms(void)
{
	TCCR2B|=(1<<CS22);
	TCCR2B|=(1<<CS21);
	TCCR2B|=(1<<CS20);
	TIMSK2|=(1<<TOIE2);
	TCNT2=TIME_20_MS;
}
void Inicia_Timer_Contador_Medio_Grado(void)
{
	TRIAC_OFF;
	TCCR0B|=(1<<CS01);
	TIMSK0|=(1<<TOIE0);
	TCNT0=TIME_MEDIO_GRADO;
}
void Detiene_Timer_Contador_Medio_Grado(void)
{
	TCCR0B&=~(1<<CS02);
	TCCR0B&=~(1<<CS01);
	TCCR0B&=~(1<<CS00);
	TIMSK0&=~(1<<TOIE0);
}
void Inicia_Interrupcion_Cruce_Cero(void)
{
	EICRA|=(1<<ISC21);
	EICRA|=(1<<ISC20);
	EIMSK|=(1<<INT2);
}
void Detiene_Interrupcion_Cruce_Cero(void)
{
	EIMSK&=~(1<<INT2);
}
void Calcula_PID(struct stVariablesPID *ptVariablesPID)
{
	if (ptVariablesPID->modo_trabajo==AUTOMATIC)
	{
		if (ptVariablesPID->tiempo_actual>=ptVariablesPID->tiempo_muestreo)
		{
			ptVariablesPID->tiempo_actual=0;
		Lee_Max31855(ptConvertidor_Int_String,ptMediciones,ptVariablesPID);
			double error=ptVariablesPID->set_point-ptVariablesPID-
>entrada;
			ptVariablesPID->termino_integral+=(ptVariablesPID->ki*error);
			if (ptVariablesPID->termino_integral>ptVariablesPID-
>salida_maxima)
```

```
                        {
                                ptVariablesPID->termino_integral=ptVariablesPID-
>salida_maxima;
                        }
                        else
                        {
                                if (ptVariablesPID->termino_integral<ptVariablesPID-
>salida_minima)
                                {
                                        ptVariablesPID-
>termino_integral=ptVariablesPID->salida_minima;
                                }
                        }
                        double d_entrada=(ptVariablesPID->entrada-ptVariablesPID-
>entrada_anterior);
                        ptVariablesPID->salida_controlador=ptVariablesPID-
>kp*error+ptVariablesPID->termino_integral-ptVariablesPID->kd*d_entrada;
                        if (ptVariablesPID->salida_controlador>ptVariablesPID-
>salida_maxima)
                        {
                                ptVariablesPID->salida_controlador=ptVariablesPID-
>salida_maxima;
                        }
                        else
                        {
                                if (ptVariablesPID->salida_controlador<ptVariablesPID-
>salida_minima)
                                {
                                        ptVariablesPID-
>salida_controlador=ptVariablesPID->salida_minima;
                                }
                        }
                        medio_grado=pgm_read_word(angulos_potencia+(unsigned
char)ptVariablesPID->salida_controlador);
                        ptVariablesPID->entrada_anterior=ptVariablesPID->entrada;
                }
        }
}
void Configura_Sintonizacion(struct stVariablesPID *ptVariablesPID,double
KP,double KI,double KD)
{
        double tiempo_muestreo_segundos=((double)ptVariablesPID-
>tiempo_muestreo*20)/1000;
        ptVariablesPID->kp=KP;
        ptVariablesPID->ki=KI*tiempo_muestreo_segundos;
        ptVariablesPID->kd=KD/tiempo_muestreo_segundos;
}
void Configura_Tiempo_Muestro(struct stVariablesPID *ptVariablesPID, unsigned int
nuevo_tiempo_muestreo)
{
        if (nuevo_tiempo_muestreo>0)
        {
                double ratio=(double)nuevo_tiempo_muestreo/((double)ptVariablesPID-
>tiempo_muestreo*20);
                ptVariablesPID->ki*=ratio;
                ptVariablesPID->kd/=ratio;
                ptVariablesPID->tiempo_muestreo=(nuevo_tiempo_muestreo/20);
        }
}
```

```
void Configura_Limites_Salida(struct stVariablesPID *ptVariablesPID,double
minimo,double maximo)
{
	if (maximo>minimo)
	{
		ptVariablesPID->salida_minima=minimo;
		ptVariablesPID->salida_maxima=maximo;

		if (ptVariablesPID->salida_controlador>ptVariablesPID-
>salida_maxima)
		{
			ptVariablesPID->salida_controlador=ptVariablesPID-
>salida_maxima;
		}
		else
		{
			if (ptVariablesPID->salida_controlador<ptVariablesPID-
>salida_minima)
			{
				ptVariablesPID->salida_controlador=ptVariablesPID-
>salida_minima;
			}
		}
		if (ptVariablesPID->termino_integral>ptVariablesPID->salida_maxima)
		{
			ptVariablesPID->termino_integral=ptVariablesPID-
>salida_maxima;
		}
		else
		{
			if (ptVariablesPID->termino_integral<ptVariablesPID-
>salida_minima)
			{
				ptVariablesPID->termino_integral=ptVariablesPID-
>salida_minima;
			}
		}
	}
}
void Configura_Modo(struct stVariablesPID *ptVariablesPID,unsigned char modo)
{
	unsigned char var=0;
	if (modo==AUTOMATIC)
	{
		var=AUTOMATIC;
	}
	if (var&& !ptVariablesPID->modo_trabajo)
	{
		Inicia_PID(ptVariablesPID);
	}
	ptVariablesPID->modo_trabajo=var;
}
void Inicia_PID(struct stVariablesPID *ptVariablesPID)
{
	ptVariablesPID->entrada_anterior=ptVariablesPID->entrada;
	ptVariablesPID->termino_integral=ptVariablesPID->salida_controlador;
	if (ptVariablesPID->termino_integral>ptVariablesPID->salida_maxima)
	{
		ptVariablesPID->termino_integral=ptVariablesPID->salida_maxima;
	}
```

```
	else
	{
		if (ptVariablesPID->termino_integral<ptVariablesPID->salida_minima)
		{
			ptVariablesPID->termino_integral=ptVariablesPID-
>salida_minima;
		}
	}
}
void Configura_SPI_MAX31855(void)
{
	SPCR|=(1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);
}
unsigned char SPI_Transferencia(unsigned char data)
{
	SPDR=data;
	while(!(SPSR & (1<<SPIF)));
	return(SPDR);
}
void Lee_Max31855(struct stConvertidorIntString *ptConvertidor_Int_String,struct
stMediciones *ptMediciones,struct stVariablesPID *ptVariablesPID)
{
	CS_MAX31855_DOWN;
	unsigned char Variable[4];
	Variable[0]=SPI_Transferencia(0);
	Variable[1]=SPI_Transferencia(0);
	Variable[2]=SPI_Transferencia(0);
	Variable[3]=SPI_Transferencia(0);
	CS_MAX31855_UP;
	unsigned int var=0;
	var=Variable[0];
	var=(var<<8);
	var=(var&0xFF00)|Variable[1];
	if (var&1)
	{
		switch ((Variable[3]&0b00000111))
		{
			case 1:	ptMediciones->parametros[TEMPERATURA]=' ';
			ptMediciones->parametros[TEMPERATURA+1]='O';
			ptMediciones->parametros[TEMPERATURA+2]='C';
			break;
			case 2:	ptMediciones->parametros[TEMPERATURA]=' ';
			ptMediciones->parametros[TEMPERATURA+1]='S';
			ptMediciones->parametros[TEMPERATURA+2]='G';
			break;
			case 4:	ptMediciones->parametros[TEMPERATURA]=' ';
			ptMediciones->parametros[TEMPERATURA+1]='S';
			ptMediciones->parametros[TEMPERATURA+2]='V';
			break;
		}
	}
	else
	{
		float var2;
		var=(var>>2)&0b0011111111111111;
		var2=(float)var*0.25;
		ptVariablesPID->entrada=(double)var2;
		unsigned int vari=0;

		vari=(int)var2;
```

```
                Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,vari);
                LCD_Goto_XY(0,1);
                if (ptConvertidor_Int_String->centenas=='0')
                {
                        ptMediciones->parametros[TEMPERATURA]=' ';
                        LCD_Write(' ');
                }
                else
                {
                        ptMediciones-
>parametros[TEMPERATURA]=ptConvertidor_Int_String->centenas;
                        LCD_Write(ptConvertidor_Int_String->centenas);
                }
                if (ptConvertidor_Int_String->centenas=='0' &&
ptConvertidor_Int_String->decenas=='0')
                {
                        ptMediciones->parametros[TEMPERATURA+1]=' ';
                        LCD_Write(' ');
                }
                else
                {
                        ptMediciones-
>parametros[TEMPERATURA+1]=ptConvertidor_Int_String->decenas;
                        LCD_Write(ptConvertidor_Int_String->decenas);
                }
                ptMediciones->parametros[TEMPERATURA+2]=ptConvertidor_Int_String-
>unidades;
                LCD_Write(ptConvertidor_Int_String->unidades);
        }
}
```

Control PID.h

```
#ifndef CONTROL_PID_H_
#define CONTROL_PID_H_
#include <avr/io.h>
#include <avr/pgmspace.h>
#include "Inicio_Micro.h"
#include "LCD2x16.h"
#define AUTOMATIC    1
#define MANUAL              0
#define CS_MAX31855  PORTB4
#define PUERTO_CS    PORTB
#define CS_MAX31855_UP      PUERTO_CS|=(1<<CS_MAX31855)
#define CS_MAX31855_DOWN    PUERTO_CS&=~(1<<CS_MAX31855)
#define PUERTO_TRIAC PORTA
#define TRIAC PORTA0
#define TRIAC_ON     PUERTO_TRIAC|=(1<<TRIAC)
#define TRIAC_OFF    PUERTO_TRIAC&=~(1<<TRIAC)
#define TIME_MEDIO_GRADO 231
#define TIME_20_MS 40
unsigned int medio_grado;
unsigned int contador_medio_grado;
void Calcula_PID(struct stVariablesPID *ptVariablesPID);
void Configura_Sintonizacion(struct stVariablesPID *ptVariablesPID,double
KP,double KI,double KD);
void Configura_Tiempo_Muestro(struct stVariablesPID *ptVariablesPID, unsigned int
nuevo_tiempo_muestreo);
void Configura_Limites_Salida(struct stVariablesPID *ptVariablesPID,double
minimo,double maximo);
```

```
void Configura_Modo(struct stVariablesPID *ptVariablesPID,unsigned char modo);
void Inicia_PID(struct stVariablesPID *ptVariablesPID);
void Configura_SPI_MAX31855(void);
void Lee_Max31855(struct stConvertidorIntString *ptConvertidor_Int_String,struct
stMediciones *ptMediciones,struct stVariablesPID *ptVariablesPID);
void Inicia_Timer_Contador_Medio_Grado(void);
void Inicia_Interrupcion_Cruce_Cero(void);
void Detiene_Timer_Contador_Medio_Grado(void);
void Inicia_Timer_Counter_20ms(void);
#endif
```

Inicio Micro.c

```
#include "Inicio_Micro.h"
void Inicia_Puertos(void)
{
	DDRC=0xFF;
	DDRB=0b10111011;
	PORTB=0b01000100;
	DDRA=0b10100001;
	PORTA=0b01011110;
	DDRD=0b10110010;
	PORTD=0b01001101;
}
void Inicia_PWM_Motor(void)
{
	TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
	TCCR1B|=(1<<WGM12)|(1<<WGM13)|(1<<CS11)|(1<<CS10);
	ICR1=172;
	OCR1A=0;
	OCR1B=0;
}
void Convierte_Int_String_Sin_Formato(struct stConvertidorIntString
*ptConvertidor_Int_String, unsigned int Numero)
{
	ptConvertidor_Int_String->d_millar=(unsigned char)(Numero/10000);
	Numero -=(unsigned int)(ptConvertidor_Int_String->d_millar*10000);
	ptConvertidor_Int_String->u_millar=(unsigned char)(Numero/1000);
	Numero -=(unsigned int)(ptConvertidor_Int_String->u_millar*1000);
	ptConvertidor_Int_String->centenas=(unsigned char)(Numero/100);
	Numero -=(unsigned int)(ptConvertidor_Int_String->centenas*100);
	ptConvertidor_Int_String->decenas=(unsigned char)(Numero/10);
	Numero -=(unsigned int)(ptConvertidor_Int_String->decenas*10);
	ptConvertidor_Int_String->unidades=(unsigned char)Numero;
	ptConvertidor_Int_String->d_millar +=0x30;
	ptConvertidor_Int_String->u_millar +=0x30;
	ptConvertidor_Int_String->centenas +=0x30;
	ptConvertidor_Int_String->decenas  +=0x30;
	ptConvertidor_Int_String->unidades +=0x30;
}
void PrintFloat(float value,int precision,char *pBuffer)
{
	int c;
	char car;
	int bufferPos=0;
	long parteEntera,parteDecimal;
	float mul10=1;
	for(c=0;c<precision;c++)
	mul10*=10;
	parteEntera=abs((long)value);
```

```
        parteDecimal=abs((long)((value-(long)value)*mul10));
        do
        {
                pBuffer[bufferPos]=(char)(parteEntera%10)+'0'; bufferPos++;
                parteEntera/=10;
        }
        while(parteEntera>0);
        if(value<0)
        {
                pBuffer[bufferPos]='-'; bufferPos++;
        }
        for(c=0;c<bufferPos/2;c++)
        {
                car=pBuffer[c]; pBuffer[c]=pBuffer[bufferPos-c-1];
pBuffer[bufferPos-c-1]=car;
        }
        if(precision>0)
        {
                pBuffer[bufferPos]='.'; bufferPos++;
                int parteDecimalPos=bufferPos;
                for(c=0;c<precision;c++)
                {
                        pBuffer[bufferPos]=(char)(parteDecimal%10)+'0'; bufferPos++;
                        parteDecimal/=10;
                }
                for(c=0;c<precision/2;c++)
                {
                        car=pBuffer[c+parteDecimalPos];
                        pBuffer[c+parteDecimalPos]=pBuffer[bufferPos-c-1];
                        pBuffer[bufferPos-c-1]=car;
                }
        }
        pBuffer[bufferPos]=0;
}
void Control_Planta(struct stMediciones *ptMediciones,struct stVariablesPID
*ptVariablesPID)
{
        if (ptMediciones->parametros[ON_PLANTA]==TRUE)
        {
                ptVariablesPID->set_point=((double)temperatura_prog*5);
                OCR1B=(velocidad_motor_24v);
                OCR1A=(velocidad_motor_15v);
                if (motor_inyector=='N')
                {
                        ON_MOTOR_INYECTORA;
                }
                else
                {
                        OFF_MOTOR_INYECTORA;
                }
                if (bit_is_set(PINA,PINA6))
                {
                        ptMediciones->parametros[INGRESO_MATERIAL]=' ';
                        ptMediciones->parametros[INGRESO_MATERIAL+1]='S';
                        ptMediciones->parametros[INGRESO_MATERIAL+2]='i';
                }
                else
                {
                        ptMediciones->parametros[INGRESO_MATERIAL]=' ';
                        ptMediciones->parametros[INGRESO_MATERIAL+1]='N';
```

```
                        ptMediciones->parametros[INGRESO_MATERIAL+2]='o';
                }
        }
        else
        {
                if (bit_is_set(PINA,PINA6))
                {
                        ptMediciones->parametros[INGRESO_MATERIAL]=' ';
                        ptMediciones->parametros[INGRESO_MATERIAL+1]='S';
                        ptMediciones->parametros[INGRESO_MATERIAL+2]='i';
                }
                else
                {
                        ptMediciones->parametros[INGRESO_MATERIAL]=' ';
                        ptMediciones->parametros[INGRESO_MATERIAL+1]='N';
                        ptMediciones->parametros[INGRESO_MATERIAL+2]='o';
                }
                ptVariablesPID->set_point=20;
                OCR1B=0;
                OCR1A=0;
                OFF_MOTOR_INYECTORA;
        }
}
```

Inicio Micro.h

```
#ifndef INICIO_MICRO_H_
#define INICIO_MICRO_H_
#include <avr/io.h>
#include <stdlib.h >
#define TRUE 1
#define FALSE 0
#define TEMPERATURA 1
#define TANQUE1 4
#define TANQUE2 7
#define INGRESO_MATERIAL 10
#define MOTOR_TRITURADORA 13
#define MOTOR_FAJA 16
#define MOTOR_INYECCION 19
#define BATERIA 25
#define ON_PLANTA 22
#define LED                 PORTA5
#define Puerto_Led PORTA
#define LED_ON      Puerto_Led|=(1<<LED)
#define LED_OFF     Puerto_Led&=~(1<<LED)
#define TOGGLE_LED Puerto_Led^=(1<<LED)
#define XBEE_PUERTO_RESET  PORTA
#define XBEE_RESET                PORTA7
#define XBEE_RESET_ON             XBEE_PUERTO_RESET&=~(1<<XBEE_RESET)
#define XBEE_RESET_OFF            XBEE_PUERTO_RESET|=(1<<XBEE_RESET)
#define FCPU 11059200
#define BAUD 9600
#define MYUBRR FCPU/16/BAUD-1
#define PUERTO_MOTOR_INYECTORA PORTD
#define MOTOR_INYECTORA PORTD7
#define      ON_MOTOR_INYECTORA
      PUERTO_MOTOR_INYECTORA|=(1<<MOTOR_INYECTORA)
#define      OFF_MOTOR_INYECTORA
      PUERTO_MOTOR_INYECTORA&=~(1<<MOTOR_INYECTORA)
#define PUERTO_MOTOR_TRITURADORA PORTD
```

```
#define MOTOR_TRITU  PORTD4
#define       ON_MOTOR_TRITURADORA
       PUERTO_MOTOR_TRITURADORA|=(1<<MOTOR_TRITU)
#define       OFF_MOTOR_TRITURADORA
       PUERTO_MOTOR_TRITURADORA&=~(1<<MOTOR_TRITU)
#define PUERTO_CALEFACTOR PORTC
#define CALEFACTOR PORTC1
#define ON_CALEFACTOR PUERTO_CALEFACTOR|=(1<<CALEFACTOR)
#define OFF_CALEFACTOR PUERTO_CALEFACTOR&=~(1<<CALEFACTOR)
struct stVariablesPID
{
       unsigned int tiempo_muestreo;
       unsigned char tiempo_actual;
       unsigned char modo_trabajo;
       double set_point;
       double error;
       double error_anterior;
       double salida_controlador;
       double entrada;
       double entrada_anterior;
       double kp;
       double ki;
       double kd;
       double termino_integral;
       double salida_maxima;
       double salida_minima;
}VariablesPID,*ptVariablesPID;
struct stConvertidorIntString
{
       unsigned char unidades;
       unsigned char decenas;
       unsigned char centenas;
       unsigned char u_millar;
       unsigned char d_millar;
}Convertidor_Int_String,*ptConvertidor_Int_String;
struct stMediciones
{
       unsigned char parametros[30];
}Mediciones,*ptMediciones;
unsigned int temperatura_prog;
unsigned char velocidad_motor_24v;
unsigned char velocidad_motor_15v;
unsigned char motor_inyector;
void Inicia_Puertos(void);
void Convierte_Int_String_Sin_Formato(struct stConvertidorIntString
*ptConvertidor_Int_String, unsigned int Numero);
void PrintFloat(float value,int precision,char *pBuffer);
void Inicia_PWM_Motor(void);
void Control_Planta(struct stMediciones *ptMediciones,struct stVariablesPID
*ptVariablesPID);
#endif
```

Módulos Ultrasónicos.c

```
#include "Modulos_Ultrasonicos.h"
void Configura_Interrupcion_Ultrasonicos(void)
{
	EICRA|=(1<<ISC10);
	EICRA|=(1<<ISC00);
}
void Activa_Interrupcion_Tolva_Grande(void)
{
	EIMSK&=~(1<<INT0);
	EIMSK|=(1<<INT1);
}
void Activa_Interrupcion_Tolva_Pequena(void)
{
	EIMSK&=~(1<<INT1);
	EIMSK|=(1<<INT0);
}
void Star_Contador(void)
{
	TCCR3B|=(1<<CS11);
	TIMSK3|=(1<<TOIE3);
	TCNT3=TC_INICIO;
}
void Stop_Contador(struct stSensoresUltrasonicos *ptSensores_Ultrasonicos,struct
stMediciones *ptMediciones,struct stConvertidorIntString
*ptConvertidor_Int_String)
{
	unsigned long var=0;
	TCCR3B=0x00;
	var=TCNT3;
	var-=TC_INICIO;
	var=((var*72)/100);
	var/=58;
	ptSensores_Ultrasonicos->contador_ocupado=FALSE;
	if (ptSensores_Ultrasonicos->flag_desborde==FALSE)
	{
		if (ptSensores_Ultrasonicos->toggle_sensores==TRUE)
		{
			ptSensores_Ultrasonicos->contador_tg=var;
	Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,ptSensores_Ultra
sonicos->contador_tg);
			if (ptConvertidor_Int_String->centenas=='0')
			{
				ptMediciones->parametros[TANQUE1]=' ';
			}
			else
			{
				ptMediciones-
>parametros[TANQUE1]=ptConvertidor_Int_String->centenas;
			}
			if (ptConvertidor_Int_String-
>centenas=='0'&&ptConvertidor_Int_String->decenas=='0')
			{
				ptMediciones->parametros[TANQUE1+1]=' ';
			}
			else
			{
				ptMediciones-
>parametros[TANQUE1+1]=ptConvertidor_Int_String->decenas;
			}
```

```
                        ptMediciones->parametros[TANQUE1+2]=ptConvertidor_Int_String-
>unidades;
                }
                else
                {
                        ptSensores_Ultrasonicos->contador_tp=var;
        Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,ptSensores_Ultra
sonicos->contador_tp);
                        if (ptConvertidor_Int_String->centenas=='0')
                        {
                                ptMediciones->parametros[TANQUE2]=' ';
                        }
                        else
                        {
                                ptMediciones-
>parametros[TANQUE2]=ptConvertidor_Int_String->centenas;
                        }
                        if (ptConvertidor_Int_String-
>centenas=='0'&&ptConvertidor_Int_String->decenas=='0')
                        {
                                ptMediciones->parametros[TANQUE2+1]=' ';
                        }
                        else
                        {
                                ptMediciones-
>parametros[TANQUE2+1]=ptConvertidor_Int_String->decenas;
                        }
                        ptMediciones->parametros[TANQUE2+2]=ptConvertidor_Int_String-
>unidades;
                }
        }
        else
        {
                ptSensores_Ultrasonicos->flag_desborde=FALSE;
        }
}
void Control_Disparos_Sensores_Ultrasonicos(struct stSensoresUltrasonicos
*ptSensores_Ultrasonicos)
{
        if (ptSensores_Ultrasonicos->contador_ocupado==FALSE)
        {
                switch (ptSensores_Ultrasonicos->toggle_sensores)
                {
                        case 0: Dispara_Sensor_Ultrasonico_Tolva_Grande();
                                ptSensores_Ultrasonicos->contador_ocupado=TRUE;
                                ptSensores_Ultrasonicos->toggle_sensores=TRUE;
                                break;
                        case 1: Dispara_Sensor_Ultrasonico_Tolva_Pequena();
                                ptSensores_Ultrasonicos->contador_ocupado=TRUE;
                                ptSensores_Ultrasonicos->toggle_sensores=FALSE;
                                ptSensores_Ultrasonicos->flag_desborde=FALSE;
                                break;
                }
        }
}
void Dispara_Sensor_Ultrasonico_Tolva_Grande(void)
{
        Activa_Interrupcion_Tolva_Grande();
        ON_TRIGGER_TG;
        _delay_us(15);
```

```
	OFF_TRIGGER_TG;
	Star_Contador();
}
void Dispara_Sensor_Ultrasonico_Tolva_Pequena(void)
{
	Activa_Interrupcion_Tolva_Pequena();
	ON_TRIGGER_TP;
	_delay_us(15);
	OFF_TRIGGER_TP;
	Star_Contador();
}
ISR(TIMER3_OVF_vect)
{
	TCCR3B=0x00;
	TCNT3=TC_INICIO;
	ptSensores_Ultrasonicos->flag_desborde=TRUE;
	ptSensores_Ultrasonicos->contador_ocupado=FALSE;
}
ISR(INT0_vect)
{
	if (bit_is_set(PIND,PIND2))
	{
		Star_Contador();
	}
	else
	{
	Stop_Contador(ptSensores_Ultrasonicos,ptMediciones,ptConvertidor_Int_Strin
g);
	}
}
ISR(INT1_vect)
{
	if (bit_is_set(PIND,PIND3))
	{
		Star_Contador();
	}
	else
	{
	Stop_Contador(ptSensores_Ultrasonicos,ptMediciones,ptConvertidor_Int_Strin
g);
	}
}
```

Módulos Ultrasónicos.h

```
#ifndef MODULOS_ULTRASONICOS_H_
#define MODULOS_ULTRASONICOS_H_
#define F_CPU 11059200UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "Inicio_Micro.h"
#define OFF_TRIGGER_TG PORTB|=(1<<PORTB1)
#define ON_TRIGGER_TG PORTB&=~(1<<PORTB1)
#define OFF_TRIGGER_TP PORTB|=(1<<PORTB0)
#define ON_TRIGGER_TP PORTB&=~(1<<PORTB0)
#define TRUE 1
#define FALSE 0
#define TC_INICIO 33446
volatile unsigned long Medicion_Tolva_Grande;
```

```
volatile unsigned int Medicion_Tolva_Pequena;
struct stSensoresUltrasonicos
{
	unsigned long contador_tg;
	unsigned long contador_tp;
	unsigned char contador_ocupado:1;
	unsigned char toggle_sensores:1;
	unsigned char flag_desborde:1;
}SensoresUltrasonicos,*ptSensores_Ultrasonicos;
void Configura_Interrupcion_Ultrasonicos(void);
void Star_Contador_Tolva_Grande(void);
void Star_Contador_Tolva_Pequena(void);
unsigned int Stop_Contador_Tolva_Pequena(void);
unsigned long Stop_Contador_Tolva_Grande(void);
void Dispara_Sensor_Ultrasonico_Tolva_Grande(void);
void Dispara_Sensor_Ultrasonico_Tolva_Pequena(void);
void Control_Disparos_Sensores_Ultrasonicos(struct stSensoresUltrasonicos
*ptSensores_Ultrasonicos);
void Control_Velocidad_Motor(struct stSensoresUltrasonicos
*ptSensores_Ultrasonicos,struct stMediciones *ptMediciones,struct
stConvertidorIntString *ptConvertidor_Int_String);
#endif
```

USART.C

```
#include "Usart.h"
void USART_Init( unsigned int ubrr)
{
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
	UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
	UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
}
ISR(USART0_TX_vect)
{
	Envia_Buffer_Tx(ptBuffer_Tx_Frames);
}
ISR(USART0_RX_vect)
{
	volatile unsigned char var=' ';
	var=UDR0;
	Llena_Buffer_Rx(ptBuffer_Rx_Frames,var);
}
void Activa_Interrupcion_Usart_Tx(void)
{
	UCSR0B|=(1<<TXCIE0);
}
void Desactiva_Interrupcion_Usart_Tx(void)
{
	UCSR0B&=~(1<<TXCIE0);
}
void Llena_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames, unsigned char
nuevo_caracter)
{
	if ((ptBuffer_Rx_Frames-
>status_buffer_1==BUFFER_LIBRE||ptBuffer_Rx_Frames-
>status_buffer_1==BUFFER_RECIBIENDO)&&ptBuffer_Rx_Frames-
>status_buffer_2!=BUFFER_RECIBIENDO&&ptBuffer_Rx_Frames-
>status_buffer_3!=BUFFER_RECIBIENDO)
	{
```

```
		if (ptBuffer_Rx_Frames->contador<BUFFER_TAMANO_MAX)
		{
			if (nuevo_caracter==INICIO_FRAME&&ptBuffer_Rx_Frames-
>status_buffer_1==BUFFER_LIBRE)
			{
				ptBuffer_Rx_Frames->status_buffer_1=BUFFER_RECIBIENDO;
			}
			if (ptBuffer_Rx_Frames->status_buffer_1==BUFFER_RECIBIENDO)
			{
				ptBuffer_Rx_Frames->buffer[ptBuffer_Rx_Frames-
>contador][BUFFER_1]=nuevo_caracter;
				ptBuffer_Rx_Frames->contador++;
				if (nuevo_caracter==FIN_FRAME)
				{
					ptBuffer_Rx_Frames-
>status_buffer_1=BUFFER_VERIFICANDO;
					ptBuffer_Rx_Frames->contador=0;
				}
			}
		}
		else
		{
			ptBuffer_Rx_Frames->contador=0;
			ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LIBRE;
		}
	}
	if ((ptBuffer_Rx_Frames-
>status_buffer_2==BUFFER_LIBRE||ptBuffer_Rx_Frames-
>status_buffer_2==BUFFER_RECIBIENDO)&&ptBuffer_Rx_Frames-
>status_buffer_1!=BUFFER_RECIBIENDO&&ptBuffer_Rx_Frames-
>status_buffer_3!=BUFFER_RECIBIENDO)
	{
		if (ptBuffer_Rx_Frames->contador<BUFFER_TAMANO_MAX)
		{
			if (nuevo_caracter==INICIO_FRAME&&ptBuffer_Rx_Frames-
>status_buffer_2==BUFFER_LIBRE)
			{
				ptBuffer_Rx_Frames->status_buffer_2=BUFFER_RECIBIENDO;
			}
			if (ptBuffer_Rx_Frames->status_buffer_2==BUFFER_RECIBIENDO)
			{
				ptBuffer_Rx_Frames->buffer[ptBuffer_Rx_Frames-
>contador][BUFFER_2]=nuevo_caracter;
				ptBuffer_Rx_Frames->contador++;
				if (nuevo_caracter==FIN_FRAME)
				{
					ptBuffer_Rx_Frames-
>status_buffer_2=BUFFER_VERIFICANDO;
					ptBuffer_Rx_Frames->contador=0;
				}
			}
		}
		else
		{
			ptBuffer_Rx_Frames->contador=0;
			ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LIBRE;
		}


	}
```

```
	if ((ptBuffer_Rx_Frames-
>status_buffer_3==BUFFER_LIBRE||ptBuffer_Rx_Frames-
>status_buffer_3==BUFFER_RECIBIENDO)&&ptBuffer_Rx_Frames-
>status_buffer_1!=BUFFER_RECIBIENDO&&ptBuffer_Rx_Frames-
>status_buffer_2!=BUFFER_RECIBIENDO)
	{
		if (ptBuffer_Rx_Frames->contador<BUFFER_TAMANO_MAX)
		{
			if (nuevo_caracter==INICIO_FRAME&&ptBuffer_Rx_Frames-
>status_buffer_3==BUFFER_LIBRE)
			{
				ptBuffer_Rx_Frames->status_buffer_3=BUFFER_RECIBIENDO;
			}
			if (ptBuffer_Rx_Frames->status_buffer_3==BUFFER_RECIBIENDO)
			{
				ptBuffer_Rx_Frames->buffer[ptBuffer_Rx_Frames-
>contador][BUFFER_3]=nuevo_caracter;
				ptBuffer_Rx_Frames->contador++;
				if (nuevo_caracter==FIN_FRAME)
				{
					ptBuffer_Rx_Frames-
>status_buffer_3=BUFFER_VERIFICANDO;
					ptBuffer_Rx_Frames->contador=0;
				}
			}
		}
		else
		{
			ptBuffer_Rx_Frames->contador=0;
			ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LIBRE;
		}
	}
}
void Verifica_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames)
{
	if (ptBuffer_Rx_Frames->status_buffer_1==BUFFER_VERIFICANDO)
	{
		unsigned char tamano_frame=0;
		unsigned char suma=0;
		tamano_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_1];

		if (ptBuffer_Rx_Frames->buffer[0][BUFFER_1]==INICIO_FRAME)
		{
			for (unsigned char xx=0;xx<=tamano_frame;xx++)
			{
				suma+=ptBuffer_Rx_Frames->buffer[xx+2][BUFFER_1];
			}
			if (suma==0xFF&&ptBuffer_Rx_Frames-
>buffer[tamano_frame+3][BUFFER_1]==FIN_FRAME)
			{
				ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LISTO;
			}
			else
			{
				ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LIBRE;
			}
		}
	}

	if (ptBuffer_Rx_Frames->status_buffer_2==BUFFER_VERIFICANDO)
```

```
	{
		unsigned char tamano_frame=0;
		unsigned char suma=0;
		tamano_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_2];
		if (ptBuffer_Rx_Frames->buffer[0][BUFFER_2]==INICIO_FRAME)
		{
			for (unsigned char xx=0;xx<=tamano_frame;xx++)
			{
				suma+=ptBuffer_Rx_Frames->buffer[xx+2][BUFFER_2];
			}
			if (suma==0xFF&&ptBuffer_Rx_Frames-
>buffer[tamano_frame+3][BUFFER_2]==FIN_FRAME)
			{
				ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LISTO;
			}
			else
			{
				ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LIBRE;
			}
		}
	}
	if (ptBuffer_Rx_Frames->status_buffer_3==BUFFER_VERIFICANDO)
	{
		unsigned char tamano_frame=0;
		unsigned char suma=0;
		tamano_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_3];

		if (ptBuffer_Rx_Frames->buffer[0][BUFFER_3]==INICIO_FRAME)
		{
			for (unsigned char xx=0;xx<=tamano_frame;xx++)
			{
				suma+=ptBuffer_Rx_Frames->buffer[xx+2][BUFFER_3];
			}
			if (suma==0xFF&&ptBuffer_Rx_Frames-
>buffer[tamano_frame+3][BUFFER_3]==FIN_FRAME)
			{
				ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LISTO;
			}
			else
			{
				ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LIBRE;
			}
		}
	}
}
void Libera_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames,struct
stMediciones *ptMediciones)
{
	if (ptBuffer_Rx_Frames->status_buffer_1==BUFFER_LISTO)
	{
		ptMediciones->parametros[ON_PLANTA]=ptBuffer_Rx_Frames-
>buffer[6][BUFFER_1];
		velocidad_motor_24v=ptBuffer_Rx_Frames->buffer[7][BUFFER_1];
		velocidad_motor_15v=ptBuffer_Rx_Frames->buffer[8][BUFFER_1];
		motor_inyector=ptBuffer_Rx_Frames->buffer[9][BUFFER_1];
		temperatura_prog=ptBuffer_Rx_Frames->buffer[10][BUFFER_1];
		ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LIBRE;
	}
	if (ptBuffer_Rx_Frames->status_buffer_2==BUFFER_LISTO)
	{
```

```
                ptMediciones->parametros[ON_PLANTA]=ptBuffer_Rx_Frames-
>buffer[6][BUFFER_2];
                velocidad_motor_24v=ptBuffer_Rx_Frames->buffer[7][BUFFER_2];
                velocidad_motor_15v=ptBuffer_Rx_Frames->buffer[8][BUFFER_2];
                motor_inyector=ptBuffer_Rx_Frames->buffer[9][BUFFER_2];
                temperatura_prog=ptBuffer_Rx_Frames->buffer[10][BUFFER_2];
                ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LIBRE;
        }
        if (ptBuffer_Rx_Frames->status_buffer_3==BUFFER_LISTO)
        {
                ptMediciones->parametros[ON_PLANTA]=ptBuffer_Rx_Frames-
>buffer[6][BUFFER_3];
                velocidad_motor_24v=ptBuffer_Rx_Frames->buffer[7][BUFFER_3];
                velocidad_motor_15v=ptBuffer_Rx_Frames->buffer[8][BUFFER_3];
                motor_inyector=ptBuffer_Rx_Frames->buffer[9][BUFFER_3];
                temperatura_prog=ptBuffer_Rx_Frames->buffer[10][BUFFER_3];
                ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LIBRE;
        }
}
void Llena_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames,struct
stMediciones *ptMediciones)
{
        if (ptBuffer_Tx_Frames->status_buffer_1==BUFFER_TX_LIBRE)
        {
                unsigned char suma=0;
                ptBuffer_Tx_Frames->buffer_tx[0]=INICIO_FRAME;
                ptBuffer_Tx_Frames->buffer_tx[1]=LARGO_FRAME;
                ptBuffer_Tx_Frames->buffer_tx[2]='M';
                ptBuffer_Tx_Frames->buffer_tx[3]='O';
                for (unsigned char rr=0;rr<LARGO_FRAME;rr++)
                {
                        ptBuffer_Tx_Frames->buffer_tx[rr+4]=ptMediciones-
>parametros[rr];
                }
                for (unsigned char bb=0; bb<LARGO_FRAME;bb++)
                {
                        suma+=ptBuffer_Tx_Frames->buffer_tx[bb+2];
                }
                suma=0xFF-suma;
                ptBuffer_Tx_Frames->buffer_tx[LARGO_FRAME+2]=suma;
                ptBuffer_Tx_Frames->buffer_tx[LARGO_FRAME+3]=FIN_FRAME;
                ptBuffer_Tx_Frames->status_buffer_1=BUFFER_TX_LISTO;
        }
}
void Inicia_Envio_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames)
{
        if (ptBuffer_Tx_Frames->status_buffer_1==BUFFER_TX_LISTO)
        {
                TOGGLE_LED;
                Activa_Interrupcion_Usart_Tx();
                ptBuffer_Tx_Frames->status_buffer_1=BUFFER_TX_ENVIANDO;
                UDR0=ptBuffer_Tx_Frames->buffer_tx[0];
                ptBuffer_Tx_Frames->contador_tx++;
        }
}
void Envia_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames)
{
        if (ptBuffer_Tx_Frames->status_buffer_1==BUFFER_TX_ENVIANDO)
        {
```

```
            if (ptBuffer_Tx_Frames->buffer_tx[ptBuffer_Tx_Frames-
>contador_tx]!=FIN_FRAME)
            {
                  UDR0=ptBuffer_Tx_Frames->buffer_tx[ptBuffer_Tx_Frames-
>contador_tx];
                  ptBuffer_Tx_Frames->contador_tx++;
            }
            else
            {
                  UDR0=ptBuffer_Tx_Frames->buffer_tx[ptBuffer_Tx_Frames-
>contador_tx];
                  ptBuffer_Tx_Frames->contador_tx=0;
                  ptBuffer_Tx_Frames->status_buffer_1=BUFFER_TX_LIBRE;
                  Desactiva_Interrupcion_Usart_Tx();
            }
      }
}
```

USART.h

```
#ifndef USART_H_
#define USART_H_
#define F_CPU 11059200UL
#include <avr/interrupt.h>
#include <avr/io.h>
#include "Inicio_Micro.h"
#define BUFFER_LIBRE 0
#define BUFFER_RECIBIENDO 1
#define BUFFER_VERIFICANDO 2
#define BUFFER_LISTO 3
#define BUFFER_1 0
#define BUFFER_2 1
#define BUFFER_3 2
#define BUFFER_TAMANO_MAX 50
#define BUFFER_CANTIDAD 3
#define TAMANO_FRAME 1
#define INICIO_FRAME 0x02
#define FIN_FRAME 0x03
#define LARGO_FRAME 27
#define BUFFER_TX_MAX 40
#define BUFFER_TX_LLENANDO 1
#define BUFFER_TX_LIBRE 0
#define BUFFER_TX_LISTO 2
#define BUFFER_TX_ENVIANDO 3
#define TRUE 1
#define FALSE 0
struct stBufferRxFrames
{
      unsigned char buffer[BUFFER_TAMANO_MAX][BUFFER_CANTIDAD];
      unsigned char status_buffer_1;
      unsigned char status_buffer_2;
      unsigned char status_buffer_3;
      unsigned char contador;
}BufferRxFrames,*ptBuffer_Rx_Frames;

struct stBufferTxFrames
{
      unsigned char buffer_tx[BUFFER_TX_MAX];
      unsigned char contador_tx;
      unsigned char status_buffer_1;
```

```
}BufferTxFrames,*ptBuffer_Tx_Frames;
void USART_Init( unsigned int ubrr);
void Llena_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames, unsigned char
nuevo_caracter);
void Verifica_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames);
void Libera_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames,struct
stMediciones *ptMediciones);
void Envia_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames);
void Llena_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames,struct
stMediciones *ptMediciones);
void Inicia_Envio_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames);
#endif
```

PLACA CONTROL DE MANDO

```
#define F_CPU 11059200UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "Inicio_Micro.h"
#include "LCD2x16.h"
#include "Interfaz.h"
volatile unsigned char Reloj=0;
void Reloj_Planta(struct stControl_LCD *ptControl_LCD)
{
	if (Reloj>=30)
	{
		Reloj=0;
		ptControl_LCD->actualizacion_lcd=TRUE;
		ptControl_LCD->actualizacion_bateria=TRUE;
	}
	else
	{
		Reloj++;
	}
}
void Inicia_Xbee(void)
{
	USART_Init (71);
	XBEE_RESET_ON;
	LED_OFF;
	LCD_Write('H');
	LCD_Write('o');
	LCD_Write('l');
	LCD_Write('a');
	LCD_Write(' ');
	LCD_Write('X');
	LCD_Write('b');
	LCD_Write('e');
	LCD_Write('e');
	LCD_Write(' ');
	LCD_Write('+');
	LCD_Write('+');
	LCD_Write('+');
	LCD_Goto_XY(0,2);
	XBEE_RESET_OFF;
	_delay_ms(500);
	_delay_ms(500);
	_delay_ms(500);
	_delay_ms(500);
```

```
sei();
Usart_Envia_Byte('+');
Usart_Envia_Byte('+');
Usart_Envia_Byte('+');
LCD_Goto_XY(0,2);
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
Usart_Envia_Byte('A');
Usart_Envia_Byte('T');
Usart_Envia_Byte('D');
Usart_Envia_Byte('H');
Usart_Envia_Byte('0');
Usart_Envia_Byte('0');
Usart_Envia_Byte('1');
Usart_Envia_Byte('3');
Usart_Envia_Byte('A');
Usart_Envia_Byte('2');
Usart_Envia_Byte('0');
Usart_Envia_Byte('0');
Usart_Envia_Byte('\r');
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
Usart_Envia_Byte('A');
Usart_Envia_Byte('T');
Usart_Envia_Byte('D');
Usart_Envia_Byte('L');
Usart_Envia_Byte('4');
Usart_Envia_Byte('0');
Usart_Envia_Byte('C');
Usart_Envia_Byte('A');
Usart_Envia_Byte('A');
Usart_Envia_Byte('E');
Usart_Envia_Byte('9');
Usart_Envia_Byte('5');
Usart_Envia_Byte('\r');
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
LCD_Goto_XY(0,2);
Usart_Envia_Byte('A');
Usart_Envia_Byte('T');
Usart_Envia_Byte('D');
Usart_Envia_Byte('L');
Usart_Envia_Byte('\r');
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
_delay_ms(500);
Usart_Envia_Byte('A');
Usart_Envia_Byte('T');
Usart_Envia_Byte('W');
Usart_Envia_Byte('R');
Usart_Envia_Byte('\r');
_delay_ms(500);
_delay_ms(500);
```

```
        _delay_ms(500);
        _delay_ms(500);
}
void Configura_xbee(void)
{
        USART_Init (71);
        XBEE_RESET_ON;
        LED_OFF;
        LCD_Write('H');
        LCD_Write('o');
        LCD_Write('l');
        LCD_Write('a');
        LCD_Write(' ');
        LCD_Write('X');
        LCD_Write('b');
        LCD_Write('e');
        LCD_Write('e');
        LCD_Write(' ');
        LCD_Write('+');
        LCD_Write('+');
        LCD_Write('+');
        LCD_Goto_XY(0,2);
        XBEE_RESET_OFF;
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        sei();
        Usart_Envia_Byte('+');
        Usart_Envia_Byte('+');
        Usart_Envia_Byte('+');
        LCD_Goto_XY(0,1);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        Usart_Envia_Byte('A');
        Usart_Envia_Byte('T');
        Usart_Envia_Byte('V');
        Usart_Envia_Byte('R');
        Usart_Envia_Byte('\r');
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        Usart_Envia_Byte('A');
        Usart_Envia_Byte('T');
        Usart_Envia_Byte('C');
        Usart_Envia_Byte('H');
        Usart_Envia_Byte('\r');
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        Usart_Envia_Byte('A');
        Usart_Envia_Byte('T');
        Usart_Envia_Byte('I');
        Usart_Envia_Byte('D');
        Usart_Envia_Byte('\r');
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        Usart_Envia_Byte('A');
        Usart_Envia_Byte('T');
```

```
		Usart_Envia_Byte('A');
		Usart_Envia_Byte('1');
		Usart_Envia_Byte('\r');
		_delay_ms(500);
		_delay_ms(500);
		_delay_ms(500);
		_delay_ms(500);



}
ISR(TIMER1_OVF_vect)
{
		Analiza_Pulsadores(ptPulsadores);
		Reloj_Planta(ptControl_LCD);
		TCNT1=TIME_PUL;
}
int main(void)
{
		_delay_ms(50);
		Inicia_Puertos();
		LCD_Inicia();
		USART_Init (71);
		XBEE_RESET_OFF;
		Configura_Interrupcion_Pulsadores();
		temperatura_set=20;
		motor_trituradora_set=8;
		motor_faja_set=8;
		motor_inyector_set='F';
		ptMediciones=&Mediciones;
		ptPulsadores=&Pulsadores;
		ptControl_LCD=&Control_LCD;
		ptControlMenus=&ControlMenus;
		ptMenu_Mediciones=&Menu_Medicioness;
		ptBuffer_Rx_Frames=&BufferRxFrames;
		ptBuffer_Tx_Frames=&BufferTxFrames;
		ptConvertidor_Int_String=&Convertidor_Int_String;
		for (unsigned char xx=0;xx<9;xx++)
		{
			ptMenu_Mediciones->contenedor_pantallas[xx]=xx;
		}
		ptControlMenus->enable_m_mediciones=TRUE;
		ptControlMenus->puntero=1;
		ptControlMenus->contador=1;
		sei();
	while(1)
	{
		Menu_Mediciones(ptMediciones,ptPulsadores,ptControlMenus,ptControl_LCD,ptM
enu_Mediciones);
		Menu_Temperatura(ptMediciones,ptPulsadores,ptControlMenus,ptControl_LCD,pt
Menu_Mediciones,ptConvertidor_Int_String);
		Menu_Motor_Trituradora_Set(ptMediciones,ptPulsadores,ptControlMenus,ptCont
rol_LCD,ptMenu_Mediciones,ptConvertidor_Int_String);
		Menu_Motor_Faja_Set(ptMediciones,ptPulsadores,ptControlMenus,ptControl_LCD
,ptMenu_Mediciones,ptConvertidor_Int_String);
		Menu_Motor_Inyector_Set(ptMediciones,ptPulsadores,ptControlMenus,ptControl
_LCD,ptMenu_Mediciones,ptConvertidor_Int_String);
			Verifica_Buffer_Rx(ptBuffer_Rx_Frames);
			Libera_Buffer_Rx(ptBuffer_Rx_Frames,ptMediciones);
			if (ptPulsadores->p_on==TRUE)
```

```
                {
                        ptPulsadores->p_on=FALSE;
                        Togle_Planta;
                }
                if (usart_responde==1)
                {
                        usart_responde=0;
                        Llena_Buffer_Tx(ptBuffer_Tx_Frames);
                        Inicia_Envio_Buffer_Tx(ptBuffer_Tx_Frames);
                }
                if (ptMediciones->parametros[ON_PLANTA]==TRUE)
                {
                        LED_ON_ENCENDIDO;
                }
                else
                {
                        LED_OFF_ENCENDIDO;
                }
                if (ptControl_LCD->actualizacion_bateria==TRUE)
                {
                        ptControl_LCD->actualizacion_bateria=FALSE;
                        unsigned int bat=0;
                        bat=lee_canal(0);
                        bat=(16*bat)/1024;
                Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,bat);
                        if (ptConvertidor_Int_String->centenas=='0')
                        {
                                ptMediciones->parametros[BATERIA]=' ';
                        }
                        else
                        {
                                ptMediciones-
>parametros[BATERIA]=ptConvertidor_Int_String->centenas;
                        }
                        if (ptConvertidor_Int_String-
>centenas=='0'&&ptConvertidor_Int_String->decenas=='0')
                        {
                                ptMediciones->parametros[BATERIA+1]=' ';
                        }
                        else
                        {
                                ptMediciones-
>parametros[BATERIA+1]=ptConvertidor_Int_String->decenas;
                        }
                        ptMediciones->parametros[BATERIA+2]=ptConvertidor_Int_String-
>unidades;
                }
        }
}
```

INICIO MICRO.c

```
#include "Inicio_Micro.h"
void Inicia_Puertos(void)
{
	DDRA=0b11111110;
	DDRD=0b10000010;
	PORTD=0b01111111;
	DDRB=0b00000011;
	DDRC=0b00000000;
}
void Convierte_Int_String_Sin_Formato(struct stConvertidorIntString
*ptConvertidor_Int_String, unsigned int Numero)
{
	ptConvertidor_Int_String->d_millar=(unsigned char)(Numero/10000);
	Numero -=(unsigned int)(ptConvertidor_Int_String->d_millar*10000);
	ptConvertidor_Int_String->u_millar=(unsigned char)(Numero/1000);
	Numero -=(unsigned int)(ptConvertidor_Int_String->u_millar*1000);
	ptConvertidor_Int_String->centenas=(unsigned char)(Numero/100);
	Numero -=(unsigned int)(ptConvertidor_Int_String->centenas*100);
	ptConvertidor_Int_String->decenas=(unsigned char)(Numero/10);
	Numero -=(unsigned int)(ptConvertidor_Int_String->decenas*10);
	ptConvertidor_Int_String->unidades=(unsigned char)Numero;
	ptConvertidor_Int_String->d_millar +=0x30;
	ptConvertidor_Int_String->u_millar +=0x30;
	ptConvertidor_Int_String->centenas +=0x30;
	ptConvertidor_Int_String->decenas  +=0x30;
	ptConvertidor_Int_String->unidades +=0x30;
}
int lee_canal(int mux)
{
	int result = 0;
	ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (0<<ADPS0);
	ADMUX = mux;
	ADMUX |= (1<<REFS0);
	ADCSRA |= (1<<ADSC);
	while (ADCSRA &(1<<ADSC))
		{
		}
	ADCSRA &= ~(1<<ADEN);
	result = ADCW;
	return result;
}
void Analiza_Pulsadores(struct stPulsadores *Pulsadores)
{
	unsigned char Actual=0;
	unsigned char Resultado=0;
	Actual=PIND;
	Actual=(Actual|0b10000111);
	Resultado=(Actual^Anterior_PuertoD);
	Resultado=(Resultado & (~(Actual)));
	Anterior_PuertoD=Actual;
	if (Resultado&PULSADOR_UP)
	{
		Pulsadores->p_up=TRUE;
	}
	if (Resultado&PULSADOR_DOWN)
	{
		Pulsadores->p_down=TRUE;
```

```
	}
	if (Resultado&PULSADOR_SET)
	{
		Pulsadores->p_set=TRUE;
	}
	if (Resultado&PULSADOR_ON)
	{
		Pulsadores->p_on=TRUE;
	}
}
void Configura_Interrupcion_Pulsadores(void)
{
	TIMSK|=(1<<TOIE1);
	TCCR1B|=(1<<CS11);
	TCNT1=TIME_PUL;
}
void USART_Init( unsigned int ubrr)
{
	UBRRH = (unsigned char)(ubrr>>8);
	UBRRL = (unsigned char)ubrr;
	UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
	UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
}
void Usart_Envia_Byte( unsigned char data )
{
    while ( !(UCSRA & (1<<UDRE)));
    UDR= data;
}
void Activa_Interrupcion_Usart_Tx(void)
{
	UCSRB|=(1<<TXCIE);
}
void Desactiva_Interrupcion_Usart_Tx(void)
{
	UCSRB&=~(1<<TXCIE);
}
ISR(USARTTXC_vect)
{
	Envia_Buffer_Tx(ptBuffer_Tx_Frames);
}
ISR(USARTRXC_vect)
{
	unsigned char var=0;
	var=UDR;
	Llena_Buffer_Rx(ptBuffer_Rx_Frames,var);
}
void Llena_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames, unsigned char
nuevo_caracter)
{
	if ((ptBuffer_Rx_Frames-
>status_buffer_1==BUFFER_LIBRE||ptBuffer_Rx_Frames-
>status_buffer_1==BUFFER_RECIBIENDO)&&ptBuffer_Rx_Frames-
>status_buffer_2!=BUFFER_RECIBIENDO&&ptBuffer_Rx_Frames-
>status_buffer_3!=BUFFER_RECIBIENDO)
	{
		TOGGLE_LED_RECEIVER;
		if (ptBuffer_Rx_Frames->contador<BUFFER_TAMANO_MAX)
		{
			if (nuevo_caracter==INICIO_FRAME&&ptBuffer_Rx_Frames-
>status_buffer_1==BUFFER_LIBRE)
```

```
                        {
                                ptBuffer_Rx_Frames->status_buffer_1=BUFFER_RECIBIENDO;
                        }
                        if (ptBuffer_Rx_Frames->status_buffer_1==BUFFER_RECIBIENDO)
                        {
                                ptBuffer_Rx_Frames->buffer[ptBuffer_Rx_Frames-
>contador][BUFFER_1]=nuevo_caracter;
                                ptBuffer_Rx_Frames->contador++;
                                if (nuevo_caracter==FIN_FRAME)
                                {
                                        ptBuffer_Rx_Frames-
>status_buffer_1=BUFFER_VERIFICANDO;
                                        ptBuffer_Rx_Frames->contador=0;
                                }
                        }
                }
                else
                {
                        ptBuffer_Rx_Frames->contador=0;
                        ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LIBRE;
                }
        }
        if ((ptBuffer_Rx_Frames-
>status_buffer_2==BUFFER_LIBRE||ptBuffer_Rx_Frames-
>status_buffer_2==BUFFER_RECIBIENDO)&&ptBuffer_Rx_Frames-
>status_buffer_1!=BUFFER_RECIBIENDO&&ptBuffer_Rx_Frames-
>status_buffer_3!=BUFFER_RECIBIENDO)
        {
                TOGGLE_LED_RECEIVER;
                if (ptBuffer_Rx_Frames->contador<BUFFER_TAMANO_MAX)
                {
                        if (nuevo_caracter==INICIO_FRAME&&ptBuffer_Rx_Frames-
>status_buffer_2==BUFFER_LIBRE)
                        {
                                ptBuffer_Rx_Frames->status_buffer_2=BUFFER_RECIBIENDO;
                        }
                        if (ptBuffer_Rx_Frames->status_buffer_2==BUFFER_RECIBIENDO)
                        {
                                ptBuffer_Rx_Frames->buffer[ptBuffer_Rx_Frames-
>contador][BUFFER_2]=nuevo_caracter;
                                ptBuffer_Rx_Frames->contador++;
                                if (nuevo_caracter==FIN_FRAME)
                                {
                                        ptBuffer_Rx_Frames-
>status_buffer_2=BUFFER_VERIFICANDO;
                                        ptBuffer_Rx_Frames->contador=0;
                                }
                        }
                }
                else
                {
                        ptBuffer_Rx_Frames->contador=0;
                        ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LIBRE;
                }
        }
```

```
	if ((ptBuffer_Rx_Frames-
>status_buffer_3==BUFFER_LIBRE||ptBuffer_Rx_Frames-
>status_buffer_3==BUFFER_RECIBIENDO)&&ptBuffer_Rx_Frames-
>status_buffer_1!=BUFFER_RECIBIENDO&&ptBuffer_Rx_Frames-
>status_buffer_2!=BUFFER_RECIBIENDO)
	{
		if (ptBuffer_Rx_Frames->contador<BUFFER_TAMANO_MAX)
		{
			if (nuevo_caracter==INICIO_FRAME&&ptBuffer_Rx_Frames-
>status_buffer_3==BUFFER_LIBRE)
			{
				ptBuffer_Rx_Frames->status_buffer_3=BUFFER_RECIBIENDO;
			}
			if (ptBuffer_Rx_Frames->status_buffer_3==BUFFER_RECIBIENDO)
			{
				ptBuffer_Rx_Frames->buffer[ptBuffer_Rx_Frames-
>contador][BUFFER_3]=nuevo_caracter;

				ptBuffer_Rx_Frames->contador++;
				if (nuevo_caracter==FIN_FRAME)
				{
					ptBuffer_Rx_Frames-
>status_buffer_3=BUFFER_VERIFICANDO;
					ptBuffer_Rx_Frames->contador=0;
				}
			}
		}
		else
		{
			ptBuffer_Rx_Frames->contador=0;
			ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LIBRE;
		}
	}
}
void Verifica_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames)
{
	if (ptBuffer_Rx_Frames->status_buffer_1==BUFFER_VERIFICANDO)
	{
		unsigned char tamano_frame=0;
		unsigned char suma=0;
		tamano_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_1];
		if (ptBuffer_Rx_Frames->buffer[0][BUFFER_1]==INICIO_FRAME)
		{
			for (unsigned char xx=0;xx<=tamano_frame;xx++)
			{
				suma+=ptBuffer_Rx_Frames->buffer[xx+2][BUFFER_1];
			}
			if (suma==0xFF&&ptBuffer_Rx_Frames-
>buffer[tamano_frame+3][BUFFER_1]==FIN_FRAME)
			{
				ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LISTO;
				usart_responde=1;
			}
			else
			{
				ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LIBRE;
			}
		}
	}
```

```
	if (ptBuffer_Rx_Frames->status_buffer_2==BUFFER_VERIFICANDO)
	{
		unsigned char tamano_frame=0;
		unsigned char suma=0;
		tamano_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_2];
		if (ptBuffer_Rx_Frames->buffer[0][BUFFER_2]==INICIO_FRAME)
		{
			for (unsigned char xx=0;xx<=tamano_frame;xx++)
			{
				suma+=ptBuffer_Rx_Frames->buffer[xx+2][BUFFER_2];
			}
			if (suma==0xFF&&ptBuffer_Rx_Frames-
>buffer[tamano_frame+3][BUFFER_2]==FIN_FRAME)
			{
				ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LISTO;
				usart_responde=1;
			}
			else
			{
				ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LIBRE;
			}
		}
	}
	if (ptBuffer_Rx_Frames->status_buffer_3==BUFFER_VERIFICANDO)
	{
		unsigned char tamano_frame=0;
		unsigned char suma=0;
		tamano_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_3];
		if (ptBuffer_Rx_Frames->buffer[0][BUFFER_3]==INICIO_FRAME)
		{
			for (unsigned char xx=0;xx<=tamano_frame;xx++)
			{
				suma+=ptBuffer_Rx_Frames->buffer[xx+2][BUFFER_3];
			}
			if (suma==0xFF&&ptBuffer_Rx_Frames-
>buffer[tamano_frame+3][BUFFER_3]==FIN_FRAME)
			{
				ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LISTO;
				usart_responde=1;
			}
			else
			{
				ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LIBRE;
			}
		}
	}
}
void Libera_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames,struct
stMediciones *ptMediciones)
{
	if (ptBuffer_Rx_Frames->status_buffer_1==BUFFER_LISTO)
	{
		unsigned char largo_frame=0;
		largo_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_1];
		for (unsigned char dd=0;dd<(largo_frame-2);dd++)
		{
			ptMediciones->parametros[dd]=ptBuffer_Rx_Frames-
>buffer[dd+4][BUFFER_1];
		}
		TOGGLE_LED_RECEIVER;
```

```
                ptBuffer_Rx_Frames->status_buffer_1=BUFFER_LIBRE;
        }
        if (ptBuffer_Rx_Frames->status_buffer_2==BUFFER_LISTO)
        {
                unsigned char largo_frame=0;
                largo_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_2];
                for (unsigned char dd=0;dd<(largo_frame-2);dd++)
                {
                        ptMediciones->parametros[dd]=ptBuffer_Rx_Frames-
>buffer[dd+4][BUFFER_2];
                }
                ptBuffer_Rx_Frames->status_buffer_2=BUFFER_LIBRE;
        }
        if (ptBuffer_Rx_Frames->status_buffer_3==BUFFER_LISTO)
        {
                unsigned char largo_frame=0;
                largo_frame=ptBuffer_Rx_Frames->buffer[TAMANO_FRAME][BUFFER_3];
                for (unsigned char dd=0;dd<(largo_frame-2);dd++)
                {
                        ptMediciones->parametros[dd]=ptBuffer_Rx_Frames-
>buffer[dd+4][BUFFER_3];
                }
                ptBuffer_Rx_Frames->status_buffer_3=BUFFER_LIBRE;
        }
}
void Llena_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames)
{
        if (ptBuffer_Tx_Frames->status_buffer_1==BUFFER_TX_LIBRE)
        {
                unsigned char suma=0;
                ptBuffer_Tx_Frames->buffer_tx[0]=INICIO_FRAME;
                ptBuffer_Tx_Frames->buffer_tx[1]=0x09;
                ptBuffer_Tx_Frames->buffer_tx[2]='S';
                ptBuffer_Tx_Frames->buffer_tx[3]='O';
                ptBuffer_Tx_Frames->buffer_tx[4]='O';
                ptBuffer_Tx_Frames->buffer_tx[5]='K';
                if (Planta_Status==TRUE)
                {
                        ptBuffer_Tx_Frames->buffer_tx[6]=TRUE;
                }
                else
                {
                        ptBuffer_Tx_Frames->buffer_tx[6]=FALSE;
                }
                ptBuffer_Tx_Frames->buffer_tx[7]=motor_trituradora_set*10;
                ptBuffer_Tx_Frames->buffer_tx[8]=motor_faja_set*10;
                ptBuffer_Tx_Frames->buffer_tx[9]=motor_inyector_set;
                ptBuffer_Tx_Frames->buffer_tx[10]=temperatura_set;
                for (unsigned char bb=0; bb<0x09;bb++)
                {
                        suma+=ptBuffer_Tx_Frames->buffer_tx[bb+2];
                }
                suma=0xFF-suma;
                ptBuffer_Tx_Frames->buffer_tx[11]=suma;
                ptBuffer_Tx_Frames->buffer_tx[12]=FIN_FRAME;
                ptBuffer_Tx_Frames->status_buffer_1=BUFFER_TX_LISTO;
        }
}
void Inicia_Envio_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames)
{
```

```
	if (ptBuffer_Tx_Frames->status_buffer_1==BUFFER_TX_LISTO)
	{
		Activa_Interrupcion_Usart_Tx();
		ptBuffer_Tx_Frames->status_buffer_1=BUFFER_TX_ENVIANDO;
		UDR=ptBuffer_Tx_Frames->buffer_tx[0];
		ptBuffer_Tx_Frames->contador_tx++;
	}
}
void Envia_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames)
{
	if (ptBuffer_Tx_Frames->status_buffer_1==BUFFER_TX_ENVIANDO)
	{
		if (ptBuffer_Tx_Frames->buffer_tx[ptBuffer_Tx_Frames-
>contador_tx]!=FIN_FRAME)
		{
			UDR=ptBuffer_Tx_Frames->buffer_tx[ptBuffer_Tx_Frames-
>contador_tx];
			ptBuffer_Tx_Frames->contador_tx++;
		}
		else
		{
			UDR=ptBuffer_Tx_Frames->buffer_tx[ptBuffer_Tx_Frames-
>contador_tx];
			ptBuffer_Tx_Frames->contador_tx=0;
			ptBuffer_Tx_Frames->status_buffer_1=BUFFER_TX_LIBRE;
			Desactiva_Interrupcion_Usart_Tx();
		}
	}
}
```

INICIO MICRO.h

```
#ifndef INICIO_MICRO_H_
#define INICIO_MICRO_H_
#define F_CPU 11059200UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include "LCD2x16.h"
#define LED                0
#define Puerto_Led PORTB
#define LED_ON       Puerto_Led|=(1<<LED)
#define LED_OFF      Puerto_Led&=~(1<<LED)
#define TOGGLE_LED_RECEIVER Puerto_Led^=(1<<LED)
#define LED_Encendido      1
#define Puerto_Led_Encendido PORTB
#define LED_ON_ENCENDIDO   Puerto_Led_Encendido|=(1<<LED_Encendido)
#define LED_OFF_ENCENDIDO  Puerto_Led_Encendido&=~(1<<LED_Encendido)
#define XBEE_PUERTO_RESET  PORTD
#define XBEE_RESET                PORTD7
#define XBEE_RESET_ON             XBEE_PUERTO_RESET&=~(1<<XBEE_RESET)
#define XBEE_RESET_OFF            XBEE_PUERTO_RESET|=(1<<XBEE_RESET)
#define FCPU 11059200
#define BAUD 9600
#define MYUBRR FCPU/16/BAUD-1
#define TRUE 1
#define FALSE 0
```

```
#define TEMPERATURA 1
#define TANQUE1 4
#define TANQUE2 7
#define INGRESO_MATERIAL 10
#define MOTOR_TRITURADORA 13
#define MOTOR_FAJA 16
#define MOTOR_INYECCION 19
#define ON_PLANTA 22
#define BATERIA 25
#define TIME_PUL 37874
#define PULSADOR_ON                64
#define PULSADOR_UP         16
#define PULSADOR_DOWN       32
#define PULSADOR_SET8
#define BUFFER_LIBRE 0
#define BUFFER_RECIBIENDO 1
#define BUFFER_VERIFICANDO 2
#define BUFFER_LISTO 3
#define BUFFER_1 0
#define BUFFER_2 1
#define BUFFER_3 2
#define BUFFER_TAMANO_MAX 60
#define BUFFER_CANTIDAD 3
#define TAMANO_FRAME 1
#define INICIO_FRAME 0x02
#define FIN_FRAME 0x03
#define BUFFER_TX_MAX 30
#define BUFFER_TX_LLENANDO 1
#define BUFFER_TX_LIBRE 0
#define BUFFER_TX_LISTO 2
#define BUFFER_TX_ENVIANDO 3
unsigned char Anterior_PuertoD;
volatile unsigned char Planta_Status;
unsigned char temperatura_set;
unsigned char motor_trituradora_set;
unsigned char motor_faja_set;
unsigned char motor_inyector_set;
unsigned char usart_responde;
#define Togle_Planta Planta_Status^=(1 << 0)
struct stConvertidorIntString
{
        unsigned char unidades;
        unsigned char decenas;
        unsigned char centenas;
        unsigned char u_millar;
        unsigned char d_millar;
}Convertidor_Int_String,*ptConvertidor_Int_String;
struct stPulsadores
{
        unsigned int p_on:1;
        unsigned int p_off:1;
        unsigned int p_manual:1;
        unsigned int p_automatico:1;
        unsigned int p_p1:1;
        unsigned int p_p2:1;
        unsigned int p_esc:1;
        unsigned int p_set:1;
        unsigned int p_up:1;
        unsigned int p_down:1;
```

```
}Pulsadores,*ptPulsadores;
struct stControl_LCD
{
	unsigned char contador;
	unsigned char actualizacion_lcd:1;
	unsigned char enable_protector_pantalla:1;
	unsigned char actualizacion_bateria:1;
}Control_LCD,*ptControl_LCD;
struct stMediciones
{
	unsigned char parametros[40];
}Mediciones,*ptMediciones;
struct stBufferRxFrames
{
	unsigned char buffer[BUFFER_TAMANO_MAX][BUFFER_CANTIDAD];
	unsigned char status_buffer_1;
	unsigned char status_buffer_2;
	unsigned char status_buffer_3;
	unsigned char contador;
}BufferRxFrames,*ptBuffer_Rx_Frames;
struct stBufferTxFrames
{
	unsigned char buffer_tx[BUFFER_TX_MAX];
	unsigned char contador_tx;
	unsigned char status_buffer_1;
}BufferTxFrames,*ptBuffer_Tx_Frames;
void Inicia_Puertos(void);
void Usart_Envia_Byte( unsigned char data );
void Configura_Interrupcion_Pulsadores(void);
void Analiza_Pulsadores(struct stPulsadores *Pulsadores);
void USART_Init( unsigned int ubrr);
void Llena_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames, unsigned char
nuevo_caracter);
void Verifica_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames);
void Libera_Buffer_Rx(struct stBufferRxFrames *ptBuffer_Rx_Frames,struct
stMediciones *ptMediciones);
void Envia_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames);
void Llena_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames);
void Inicia_Envio_Buffer_Tx(struct stBufferTxFrames *ptBuffer_Tx_Frames);
int lee_canal(int mux);
void Convierte_Int_String_Sin_Formato(struct stConvertidorIntString
*ptConvertidor_Int_String, unsigned int Numero);
#endif
```

INTERFAZ.h

```
#include "Interfaz.h"
void Pantalla_Saludo(unsigned char x,unsigned char y)
{
	LCD_Goto_XY(x,y);
	LCD_Write('C');
	LCD_Write('a');
	LCD_Write('r');
	LCD_Write('l');
	LCD_Write('o');
	LCD_Write('s');
	LCD_Write('V');
	LCD_Write('i');
```

```
        LCD_Write('l');
        LCD_Write('c');
        LCD_Write('h');
        LCD_Write('e');
        LCD_Write('r');
        LCD_Write('r');
        LCD_Write('e');
        LCD_Write('z');
}
void Pantalla_Temperatura(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
        LCD_Goto_XY(x,y);
        if (puntero==TRUE)
        {
                LCD_Write(0b01111110);
        }
        else
        {
                LCD_Write(' ');
        }
        LCD_Write('T');
        LCD_Write('e');
        LCD_Write('m');
        LCD_Write('p');
        LCD_Write('e');
        LCD_Write('r');
        LCD_Write('a');
        LCD_Write(' ');
        LCD_Write(':');
        LCD_Write(' ');
        LCD_Write(ptMediciones->parametros[TEMPERATURA]);
        LCD_Write(ptMediciones->parametros[TEMPERATURA+1]);
        LCD_Write(ptMediciones->parametros[TEMPERATURA+2]);
        LCD_Write(0b11011111);
        LCD_Write('C');
}
void Pantalla_Tanque_1(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
        LCD_Goto_XY(x,y);
        if (puntero==TRUE)
        {
                LCD_Write(0b01111110);
        }
        else
        {
                LCD_Write(' ');
        }
        LCD_Write('T');
        LCD_Write('a');
        LCD_Write('n');
        LCD_Write('q');
        LCD_Write('u');
        LCD_Write('e');
        LCD_Write('1');
        LCD_Write(' ');
        LCD_Write(':');
        LCD_Write(' ');
        LCD_Write(ptMediciones->parametros[TANQUE1]);
```

```
	LCD_Write(ptMediciones->parametros[TANQUE1+1]);
	LCD_Write(ptMediciones->parametros[TANQUE1+2]);
	LCD_Write('c');
	LCD_Write('m');
}
void Pantalla_Tanque_2(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
	LCD_Goto_XY(x,y);
	if (puntero==TRUE)
	{
		LCD_Write(0b01111110);
	}
	else
	{
		LCD_Write(' ');
	}
	LCD_Write('T');
	LCD_Write('a');
	LCD_Write('n');
	LCD_Write('q');
	LCD_Write('u');
	LCD_Write('e');
	LCD_Write('2');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	LCD_Write(ptMediciones->parametros[TANQUE2]);
	LCD_Write(ptMediciones->parametros[TANQUE2+1]);
	LCD_Write(ptMediciones->parametros[TANQUE2+2]);
	LCD_Write('c');
	LCD_Write('m');
}
void Pantalla_Ingreso_Material(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
	LCD_Goto_XY(x,y);
	if (puntero==TRUE)
	{
		LCD_Write(0b01111110);
	}
	else
	{
		LCD_Write(' ');
	}
	LCD_Write('I');
	LCD_Write('n');
	LCD_Write(' ');
	LCD_Write('P');
	LCD_Write('l');
	LCD_Write('a');
	LCD_Write('s');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	LCD_Write(ptMediciones->parametros[INGRESO_MATERIAL]);
	LCD_Write(ptMediciones->parametros[INGRESO_MATERIAL+1]);
	LCD_Write(ptMediciones->parametros[INGRESO_MATERIAL+2]);
	LCD_Write(' ');
	LCD_Write(' ');
```

```
}
void Pantalla_Motor_Trituradora(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
	LCD_Goto_XY(x,y);
	if (puntero==TRUE)
	{
		LCD_Write(0b01111110);
	}
	else
	{
		LCD_Write(' ');
	}
	LCD_Write('M');
	LCD_Write('.');
	LCD_Write('T');
	LCD_Write('r');
	LCD_Write('i');
	LCD_Write('t');
	LCD_Write('u');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,motor_triturador
a_set);
	if (ptConvertidor_Int_String->centenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->centenas);
	}
	if (ptConvertidor_Int_String->centenas=='0' && ptConvertidor_Int_String-
>decenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->decenas);
	}
	LCD_Write(ptConvertidor_Int_String->unidades);
	LCD_Write(' ');
	LCD_Write('p');
}

void Pantalla_Motor_Faja(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
	LCD_Goto_XY(x,y);
	if (puntero==TRUE)
	{
		LCD_Write(0b01111110);
	}
	else
	{
		LCD_Write(' ');
	}
	LCD_Write('M');
```

```
	LCD_Write('.');
	LCD_Write('F');
	LCD_Write('a');
	LCD_Write('j');
	LCD_Write('a');
	LCD_Write(' ');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,motor_faja_set);
	if (ptConvertidor_Int_String->centenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->centenas);
	}
	if (ptConvertidor_Int_String->centenas=='0' && ptConvertidor_Int_String-
>decenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->decenas);
	}
	LCD_Write(ptConvertidor_Int_String->unidades);
	LCD_Write(' ');
	LCD_Write('p');
}
void Pantalla_Motor_Inyectora(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
	LCD_Goto_XY(x,y);
	if (puntero==TRUE)
	{
		LCD_Write(0b01111110);
	}
	else
	{
		LCD_Write(' ');
	}
	LCD_Write('M');
	LCD_Write('.');
	LCD_Write('I');
	LCD_Write('n');
	LCD_Write('y');
	LCD_Write('e');
	LCD_Write('c');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	if (motor_inyector_set=='N')
	{
		LCD_Write(' ');
		LCD_Write('O');
		LCD_Write('N');
	}
	else
```

```
	{
		LCD_Write('O');
		LCD_Write('F');
		LCD_Write('F');
	}
	LCD_Write(' ');
	LCD_Write(' ');
}
void Pantalla_Bateria(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones)
{
	LCD_Goto_XY(x,y);
	if (puntero==TRUE)
	{
		LCD_Write(0b01111110);
	}
	else
	{
		LCD_Write(' ');
	}
	LCD_Write('B');
	LCD_Write('a');
	LCD_Write('t');
	LCD_Write('e');
	LCD_Write('r');
	LCD_Write('i');
	LCD_Write('a');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	LCD_Write(ptMediciones->parametros[BATERIA]);
	LCD_Write(ptMediciones->parametros[BATERIA+1]);
	LCD_Write(ptMediciones->parametros[BATERIA+2]);
	LCD_Write(' ');
	LCD_Write('V');
}
void Ubica_Puntero(struct stControlMenus *ptControlMenus)
{
	switch (ptControlMenus->puntero)
	{
		case 0:		ptControlMenus->linea1=FALSE;
				ptControlMenus->linea2=FALSE;
				ptControlMenus->linea3=FALSE;
				ptControlMenus->linea4=FALSE;
				break;
		case 1:		ptControlMenus->linea1=TRUE;
				ptControlMenus->linea2=FALSE;
				ptControlMenus->linea3=FALSE;
				ptControlMenus->linea4=FALSE;
				break;
		case 2:		ptControlMenus->linea1=FALSE;
				ptControlMenus->linea2=TRUE;
				ptControlMenus->linea3=FALSE;
				ptControlMenus->linea4=FALSE;
				break;
				break;
		default:ptControlMenus->linea1=TRUE;
				ptControlMenus->linea2=FALSE;
				ptControlMenus->linea3=FALSE;
				ptControlMenus->linea4=FALSE;
```

```
	}
}
void Retorna_Pantallas(unsigned char x,unsigned char y, unsigned char
puntero,unsigned char numero_pantalla,struct stMediciones *ptMediciones)
{
	switch (numero_pantalla)
	{
		case 0:		Pantalla_Saludo(x,y);
				break;
		case 1:		Pantalla_Temperatura(x,y,puntero,ptMediciones);
				break;
		case 2:		Pantalla_Tanque_1(x,y,puntero,ptMediciones);
				break;
		case 3:		Pantalla_Tanque_2(x,y,puntero,ptMediciones);
				break;
		case 4:		Pantalla_Ingreso_Material(x,y,puntero,ptMediciones);
				break;
		case 5:		Pantalla_Motor_Trituradora(x,y,puntero,ptMediciones);
				break;
		case 6:		Pantalla_Motor_Faja(x,y,puntero,ptMediciones);
				break;
		case 7:		Pantalla_Motor_Inyectora(x,y,puntero,ptMediciones);
				break;
		case 8:		Pantalla_Bateria(x,y,puntero,ptMediciones);
				break;
		default:Pantalla_Saludo(x,y);
	}
}
void Menu_Mediciones(struct stMediciones *ptMediciones,struct stPulsadores
*ptPulsadores,struct stControlMenus *ptControlMenus,struct stControl_LCD
*ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones)
{
	if(ptControlMenus->enable_m_mediciones==TRUE)
	{
		if (ptPulsadores->p_up==TRUE)
		{
			ptPulsadores->p_up=FALSE;
			ptControl_LCD->actualizacion_lcd=TRUE;
			ptControlMenus->puntero--;
			if (ptControlMenus->puntero<1)
			{
				ptControlMenus->puntero=1;
				ptControlMenus->contador--;
				if (ptControlMenus->contador<1)
				{
					ptControlMenus->contador=1;
				}
			}
		}
		if (ptPulsadores->p_down==TRUE)
		{
			ptPulsadores->p_down=FALSE;
			ptControl_LCD->actualizacion_lcd=TRUE;
			ptControlMenus->puntero++;
			if (ptControlMenus->puntero>2)
			{
				ptControlMenus->puntero=2;
				ptControlMenus->contador++;
				if (ptControlMenus->contador>7)
				{
```

```
                                        ptControlMenus->contador=7;
                                }
                        }
                }
                if (ptPulsadores->p_set==TRUE)
                {
                        ptPulsadores->p_set=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        Decisiones_Pulsador_Set(ptControlMenus);
                }
                if (ptControl_LCD->actualizacion_lcd==TRUE&&ptControlMenus-
>enable_m_mediciones==TRUE)
                {
                        ptControl_LCD->actualizacion_lcd=FALSE;
                        Ubica_Puntero(ptControlMenus);
                        Retorna_Pantallas(0,1,ptControlMenus-
>linea1,ptMenu_Mediciones->contenedor_pantallas[ptControlMenus-
>contador],ptMediciones);
                        Retorna_Pantallas(0,2,ptControlMenus-
>linea2,ptMenu_Mediciones->contenedor_pantallas[ptControlMenus-
>contador+1],ptMediciones);
                }
        }
}
void Decisiones_Pulsador_Set(struct stControlMenus *ptControlMenus)
{
        unsigned char casos=0;

        if (ptControlMenus->puntero==1 && ptControlMenus->contador==1)
        {
                casos=1;
        }
        if ((ptControlMenus->puntero==1 && ptControlMenus-
>contador==5)||(ptControlMenus->puntero==2 && ptControlMenus->contador==4))
        {
                casos=2;
        }
        if ((ptControlMenus->puntero==1 && ptControlMenus->contador==6) ||
(ptControlMenus->puntero==2 && ptControlMenus->contador==5))
        {
                casos=3
        }
        if ((ptControlMenus->puntero==1 && ptControlMenus->contador==7) ||
(ptControlMenus->puntero==2 && ptControlMenus->contador==6))
        {
                casos=4;
        }
        switch (casos)
        {
                case 1:       ptControlMenus->enable_m_mediciones=FALSE;
                              ptControlMenus->enable_m_temperatura=TRUE;
                              LCD_Send_Command(CLR_DISP);
                              break;
                case 2: ptControlMenus->enable_m_mediciones=FALSE;
                              ptControlMenus->enable_m_motor_trituradora=TRUE;
                              LCD_Send_Command(CLR_DISP);
                              break;
                case 3: ptControlMenus->enable_m_mediciones=FALSE;
                              ptControlMenus->enable_m_motor_faja=TRUE;
                              LCD_Send_Command(CLR_DISP);
```

```
                        break;
                case 4:         ptControlMenus->enable_m_mediciones=FALSE;
                                ptControlMenus->enable_m_motor_inyector=TRUE;
                                LCD_Send_Command(CLR_DISP);
                                break;
        }
}
void Menu_Temperatura(struct stMediciones *ptMediciones,struct stPulsadores
*ptPulsadores,struct stControlMenus *ptControlMenus,struct stControl_LCD
*ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String)
{
        if (ptControlMenus->enable_m_temperatura==TRUE)
        {
                if (ptPulsadores->p_up==TRUE)
                {
                        ptPulsadores->p_up=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        temperatura_set++;
                        if (temperatura_set>60)
                        {
                                temperatura_set=60;
                        }
                }
                if (ptPulsadores->p_down==TRUE)
                {
                        ptPulsadores->p_down=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        temperatura_set--;
                        if (temperatura_set<4)
                        {
                                temperatura_set=4;
                        }
                }
                if (ptPulsadores->p_set==TRUE)
                {
                        ptPulsadores->p_set=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        ptControlMenus->enable_m_temperatura=FALSE;
                        ptControlMenus->enable_m_mediciones=TRUE;
                }
                if (ptControl_LCD->actualizacion_lcd==TRUE&&ptControlMenus-
>enable_m_temperatura==TRUE)
                {
                        ptControl_LCD->actualizacion_lcd=FALSE;
                        Pantalla_Temperatura_Set(ptConvertidor_Int_String);
                }
        }
}
void Menu_Motor_Faja_Set(struct stMediciones *ptMediciones,struct stPulsadores
*ptPulsadores,struct stControlMenus *ptControlMenus,struct stControl_LCD
*ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String)
{
        if (ptControlMenus->enable_m_motor_faja==TRUE)
        {
                if (ptPulsadores->p_up==TRUE)
                {
                        ptPulsadores->p_up=FALSE;
```

```
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        motor_faja_set++;
                        if (motor_faja_set>17)
                        {
                                motor_faja_set=17;
                        }
                }
                if (ptPulsadores->p_down==TRUE)
                {
                        ptPulsadores->p_down=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        motor_faja_set--;
                        if (motor_faja_set<1)
                        {
                                motor_faja_set=1;
                        }
                }
                if (ptPulsadores->p_set==TRUE)
                {
                        ptPulsadores->p_set=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        ptControlMenus->enable_m_motor_faja=FALSE;
                        ptControlMenus->enable_m_mediciones=TRUE;
                }
                if (ptControl_LCD->actualizacion_lcd==TRUE&&ptControlMenus-
>enable_m_motor_faja==TRUE)
                {
                        ptControl_LCD->actualizacion_lcd=FALSE;
                        Pantalla_Motor_Faja_Set(ptConvertidor_Int_String);
                }
        }
}
void Menu_Motor_Trituradora_Set(struct stMediciones *ptMediciones,struct
stPulsadores *ptPulsadores,struct stControlMenus *ptControlMenus,struct
stControl_LCD *ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String)
{
        if (ptControlMenus->enable_m_motor_trituradora==TRUE)
        {
                if (ptPulsadores->p_up==TRUE)
                {
                        ptPulsadores->p_up=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        motor_trituradora_set++;
                        if (motor_trituradora_set>17)
                        {
                                motor_trituradora_set=17;
                        }
                }
                if (ptPulsadores->p_down==TRUE)
                {
                        ptPulsadores->p_down=FALSE;
                        ptControl_LCD->actualizacion_lcd=TRUE;
                        motor_trituradora_set--;
                        if (motor_trituradora_set<1)
                        {
                                motor_trituradora_set=1;
                        }
                }
                if (ptPulsadores->p_set==TRUE)
```

```
		{
			ptPulsadores->p_set=FALSE;
			ptControl_LCD->actualizacion_lcd=TRUE;
			ptControlMenus->enable_m_motor_trituradora=FALSE;
			ptControlMenus->enable_m_mediciones=TRUE;
		}
		if (ptControl_LCD->actualizacion_lcd==TRUE&&ptControlMenus-
>enable_m_motor_trituradora==TRUE)
		{
			ptControl_LCD->actualizacion_lcd=FALSE;
			Pantalla_Motor_Trituradora_Set(ptConvertidor_Int_String);
		}
	}
}
void Menu_Motor_Inyector_Set(struct stMediciones *ptMediciones,struct
stPulsadores *ptPulsadores,struct stControlMenus *ptControlMenus,struct
stControl_LCD *ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String)
{
	if (ptControlMenus->enable_m_motor_inyector==TRUE)
	{
		if (ptPulsadores->p_up==TRUE)
		{
			ptPulsadores->p_up=FALSE;
			ptControl_LCD->actualizacion_lcd=TRUE;

			if (motor_inyector_set=='N')
			{
				motor_inyector_set='F';
			}
			else
			{
				motor_inyector_set='N';
			}
		}
		if (ptPulsadores->p_down==TRUE)
		{
			ptPulsadores->p_down=FALSE;
			ptControl_LCD->actualizacion_lcd=TRUE;
			if (motor_inyector_set=='N')
			{
				motor_inyector_set='F';
			}
			else
			{
				motor_inyector_set='N';
			}
		}
		if (ptPulsadores->p_set==TRUE)
		{
			ptPulsadores->p_set=FALSE;
			ptControl_LCD->actualizacion_lcd=TRUE;
			ptControlMenus->enable_m_motor_inyector=FALSE;
			ptControlMenus->enable_m_mediciones=TRUE;

			motor_inyector_set='F';
		}
		if (ptControl_LCD->actualizacion_lcd==TRUE&&ptControlMenus-
>enable_m_motor_inyector==TRUE)
		{
```

```
                        ptControl_LCD->actualizacion_lcd=FALSE;
                        Pantalla_Motor_Inyector_Set(ptConvertidor_Int_String);
                }
        }
}
void Pantalla_Motor_Inyector_Set(struct stConvertidorIntString
*ptConvertidor_Int_String)
{
        LCD_Goto_XY(0,1);
        LCD_Write(0b01111110);
        LCD_Write('S');
        LCD_Write('e');
        LCD_Write('t');
        LCD_Write(' ');
        LCD_Write('M');
        LCD_Write('I');
        LCD_Write('n');
        LCD_Write(' ');
        LCD_Write(':');
        LCD_Write(' ');
        if (motor_inyector_set=='N')
        {
                LCD_Write(' ');
                LCD_Write('O');
                LCD_Write('N');
        }
        else
        {
                LCD_Write('O');
                LCD_Write('F');
                LCD_Write('F');
        }
        LCD_Write(' ');
        LCD_Write(' ');
}
void Pantalla_Motor_Faja_Set(struct stConvertidorIntString
*ptConvertidor_Int_String)
{
        LCD_Goto_XY(0,1);
        LCD_Write(0b01111110);
        LCD_Write('S');
        LCD_Write('e');
        LCD_Write('t');
        LCD_Write(' ');
        LCD_Write('M');
        LCD_Write('F');
        LCD_Write('a');
        LCD_Write(' ');
        LCD_Write(':');
        LCD_Write(' ');
        Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,motor_faja_set);
        if (ptConvertidor_Int_String->centenas=='0')
        {
                LCD_Write(' ');
        }
        else
        {
                LCD_Write(ptConvertidor_Int_String->centenas);
        }
```

```
	if (ptConvertidor_Int_String->centenas=='0' && ptConvertidor_Int_String-
>decenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->decenas);
	}
	LCD_Write(ptConvertidor_Int_String->unidades);
	LCD_Write('p');
	LCD_Write(' ');
}
void Pantalla_Motor_Trituradora_Set(struct stConvertidorIntString
*ptConvertidor_Int_String)
{
	LCD_Goto_XY(0,1);
	LCD_Write(0b01111110);
	LCD_Write('S');
	LCD_Write('e');
	LCD_Write('t');
	LCD_Write(' ');
	LCD_Write('M');
	LCD_Write('T');
	LCD_Write('r');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,motor_triturador
a_set);
	if (ptConvertidor_Int_String->centenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->centenas);
	}
	if (ptConvertidor_Int_String->centenas=='0' && ptConvertidor_Int_String-
>decenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->decenas);
	}
	LCD_Write(ptConvertidor_Int_String->unidades);
	LCD_Write('p');
	LCD_Write(' ');
}
void Pantalla_Temperatura_Set(struct stConvertidorIntString
*ptConvertidor_Int_String)
{
	unsigned int var=0;
	LCD_Goto_XY(0,1);
	LCD_Write(0b01111110);
	LCD_Write('S');
	LCD_Write('e');
	LCD_Write('t');
```

```
	LCD_Write(' ');
	LCD_Write('T');
	LCD_Write('e');
	LCD_Write('m');
	LCD_Write(' ');
	LCD_Write(':');
	LCD_Write(' ');
	var=temperatura_set*5;
	Convierte_Int_String_Sin_Formato(ptConvertidor_Int_String,var);
	if (ptConvertidor_Int_String->centenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->centenas);
	}
	if (ptConvertidor_Int_String->centenas=='0' && ptConvertidor_Int_String-
>decenas=='0')
	{
		LCD_Write(' ');
	}
	else
	{
		LCD_Write(ptConvertidor_Int_String->decenas);
	}
	LCD_Write(ptConvertidor_Int_String->unidades);
	LCD_Write(0b11011111);
	LCD_Write('C');
}
```

INTERFAZ.h

```
#ifndef INTERFAZ_H_
#define INTERFAZ_H_
#define F_CPU 11059200UL
#include "Inicio_Micro.h"
#include "LCD2x16.h"
struct stControlMenus
{
	unsigned char control_puntero_alarma;
	unsigned char contador;
	unsigned char puntero;
	unsigned int enable_m_principal:1;
	unsigned int enable_m_configuracion:1;
	unsigned int enable_m_potencias:1;
	unsigned int enable_m_mediciones:1;
	unsigned int enable_m_motor_inyector:1;
	unsigned int enable_m_motor_faja:1;
	unsigned int enable_m_motor_trituradora:1;
	unsigned int enable_m_temperatura:1;
	unsigned int linea1:1;
	unsigned int linea2:1;
	unsigned int linea3:1;
	unsigned int linea4:1;
	unsigned int parpadeo_puntero:1;
	unsigned int toggle:1;
}ControlMenus,*ptControlMenus;
```

```
struct stMenuMediciones
{
	unsigned char contenedor_pantallas[11];
	unsigned char medicion_seleccionada;
}Menu_Medicioness,*ptMenu_Mediciones;
void Pantalla_Temperatura(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Tanque_1(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Tanque_2(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Ingreso_Material(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Motor_Trituradora(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Motor_Faja(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Motor_Inyectora(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Pantalla_Bateria(unsigned char x,unsigned char y,unsigned char
puntero,struct stMediciones *ptMediciones);
void Menu_Mediciones(struct stMediciones *ptMediciones,struct stPulsadores
*ptPulsadores,struct stControlMenus *ptControlMenus,struct stControl_LCD
*ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones);
void Pantalla_Temperatura_Set(struct stConvertidorIntString
*ptConvertidor_Int_String);
void Decisiones_Pulsador_Set(struct stControlMenus *ptControlMenus);
void Menu_Temperatura(struct stMediciones *ptMediciones,struct stPulsadores
*ptPulsadores,struct stControlMenus *ptControlMenus,struct stControl_LCD
*ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String);
void Pantalla_Motor_Trituradora_Set(struct stConvertidorIntString
*ptConvertidor_Int_String);
void Menu_Motor_Trituradora_Set(struct stMediciones *ptMediciones,struct
stPulsadores *ptPulsadores,struct stControlMenus *ptControlMenus,struct
stControl_LCD *ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String);
void Menu_Motor_Faja_Set(struct stMediciones *ptMediciones,struct stPulsadores
*ptPulsadores,struct stControlMenus *ptControlMenus,struct stControl_LCD
*ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String);
void Pantalla_Motor_Faja_Set(struct stConvertidorIntString
*ptConvertidor_Int_String);
void Menu_Motor_Inyector_Set(struct stMediciones *ptMediciones,struct
stPulsadores *ptPulsadores,struct stControlMenus *ptControlMenus,struct
stControl_LCD *ptControl_LCD,struct stMenuMediciones *ptMenu_Mediciones,struct
stConvertidorIntString *ptConvertidor_Int_String);
void Pantalla_Motor_Inyector_Set(struct stConvertidorIntString
*ptConvertidor_Int_String);
#endif
```

LCD2X16.c

```
#include "LCD2x16.h"
void LCD_Inicia(void)
{
        _delay_ms(15);
        LCD_RS_OFF;
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_ON;
        LCD_D4_ON;
        LCD_EN_OFF;
        _delay_ms(5);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_ON;
        LCD_D4_ON;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_ON;
        LCD_D4_ON;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_ON;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_ON;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_ON;
        LCD_D6_OFF;
        LCD_D5_OFF;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_OFF;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_ON;
        LCD_D6_ON;
        LCD_D5_OFF;
```

```
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_OFF;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_OFF;
        LCD_D4_ON;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_OFF;
        LCD_D5_OFF;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_us(500);
        LCD_EN_ON;
        LCD_D7_OFF;
        LCD_D6_ON;
        LCD_D5_ON;
        LCD_D4_OFF;
        LCD_EN_OFF;
        _delay_ms(1);
}
void LCD_Write( char caracter )
{
        unsigned char Var=0;
        LCD_RS_ON;
        LCD_EN_ON;
        Var=caracter&128;
        if (Var==128)
        {
                LCD_D7_ON;
        }
        else
        {
                LCD_D7_OFF;
        }
        Var=caracter&64;
        if (Var==64)
        {
                LCD_D6_ON;
        }
        else
        {
                LCD_D6_OFF;
        }
        Var=caracter&32;
        if (Var==32)
        {
                LCD_D5_ON;
        }
```

```
        else
        {
                LCD_D5_OFF;
        }
        Var=caracter&16;
        if (Var==16)
        {
                LCD_D4_ON;
        }
        else
        {
                LCD_D4_OFF;
        }
        LCD_EN_OFF;
        _delay_us(50);
        LCD_EN_ON;
        Var=caracter&8;
        if (Var==8)
        {
                LCD_D7_ON;
        }
        else
        {
                LCD_D7_OFF;
        }
        Var=caracter&4;
        if (Var==4)
        {
                LCD_D6_ON;
        }
        else
        {
                LCD_D6_OFF;
        }
        Var=caracter&2;
        if (Var==2)
        {
                LCD_D5_ON;
        }
        else
        {
                LCD_D5_OFF;
        }
        Var=caracter&1;
        if (Var==1)
        {
                LCD_D4_ON;
        }
        else
        {
                LCD_D4_OFF;
        }
        LCD_EN_OFF;
        _delay_us(50);
}
void LCD_Send_Command( int comando )
{
        unsigned char Var=0;
        LCD_RS_OFF;
        LCD_EN_ON;
```

```
Var=comando&128;
if (Var==128)
{
	LCD_D7_ON;
}
else
{
	LCD_D7_OFF;
}
Var=comando&64;
if (Var==64)
{
	LCD_D6_ON;
}
else
{
	LCD_D6_OFF;
}
Var=comando&32;
if (Var==32)
{
	LCD_D5_ON;
}
else
{
	LCD_D5_OFF;
}
Var=comando&16;
if (Var==16)
{
	LCD_D4_ON;
}
else
{
	LCD_D4_OFF;
}
LCD_EN_OFF;
_delay_us(50);
LCD_EN_ON;
Var=comando&8;
if (Var==8)
{
	LCD_D7_ON;
}
else
{
	LCD_D7_OFF;
}
Var=comando&4;
if (Var==4)
{
	LCD_D6_ON;
}
else
{
	LCD_D6_OFF;
}
Var=comando&2;
if (Var==2)
{
```

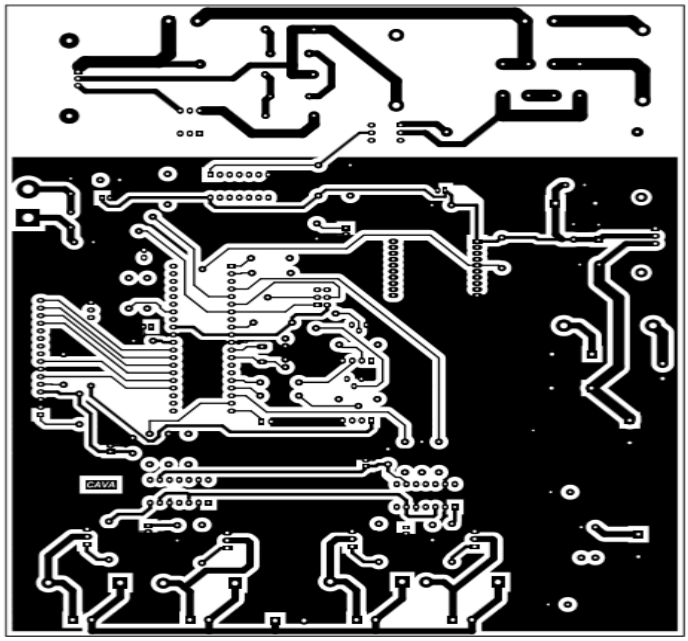
```
                LCD_D5_ON;
        }
        else
        {
                LCD_D5_OFF;
        }
        Var=comando&1;
        if (Var==1)
        {
                LCD_D4_ON;
        }
        else
        {
                LCD_D4_OFF;
        }
        LCD_EN_OFF;
        _delay_ms(3);
}
void LCD_Goto_XY( unsigned char X, unsigned char Y )
{
        unsigned char DDRAMAdddr;
        switch(Y)
        {
                case 1: DDRAMAdddr = 00 + X; break;
                case 2: DDRAMAdddr = 64 + X; break;
                default: DDRAMAdddr = 00+ X;
        }
        LCD_Send_Command(DDRAMAdddr | 0b10000000);
}
void LCD_String( char* data, int NBytes )
{
        int dato;
        if(!data)return;
        for(dato=0; dato<NBytes; dato++)
        {
                LCD_Write(data[dato]);
        }
}
void LCD_New_Caracter(unsigned int ubicacion, char* caracter)
{
        if(ubicacion<8)
        {
                LCD_Send_Command(0x40+(ubicacion*8));
                for(int l=0;l<8;l++)
                {
                        LCD_Write(caracter[l]);
                }
        }
}
```

LCD2X16.h

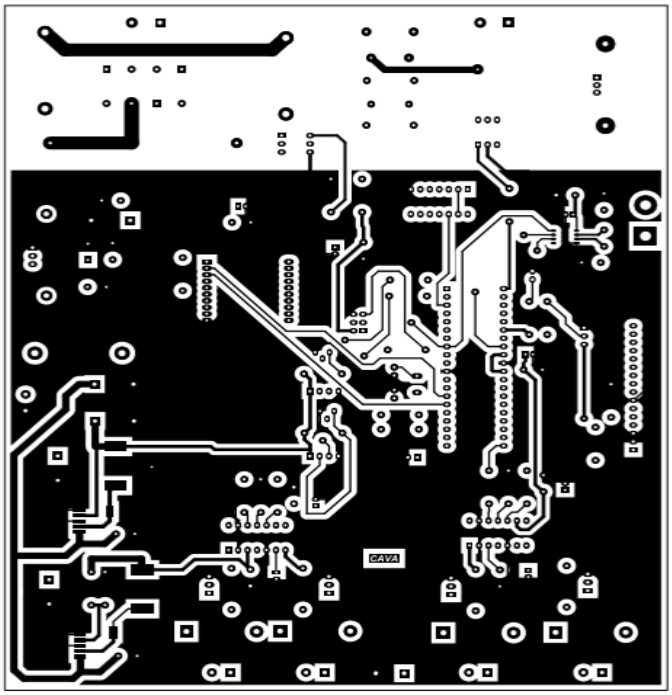
```
#ifndef LCD2X16_H_
#define LCD2X16_H_
#define F_CPU 11059200UL
#include <util/delay.h>
#include <avr/io.h>
#define PortData     PORTA
#define PortRS              PORTA
#define PortEN              PORTA
#define PortD7              PORTA
#define PortD6              PORTA
#define PortD5              PORTA
#define PortD4              PORTA
#define LCD_RS       7
#define LCD_E 6
#define LCD_D4       5
#define LCD_D5       4
#define LCD_D6       3
#define LCD_D7       2
#define BACKLIGH 7
#define Puerto_BACKLIGH PORTD
#define  LCD_RS_ON   PortRS|=(1<<LCD_RS)
#define  LCD_EN_ON   PortEN|=(1<<LCD_E)
#define LCD_D7_ON    PortD7|=(1<<LCD_D7)
#define LCD_D6_ON    PortD6|=(1<<LCD_D6)
#define LCD_D5_ON    PortD5|=(1<<LCD_D5)
#define LCD_D4_ON    PortD4|=(1<<LCD_D4)
#define LCD_RS_OFF   PortRS&=~(1<<LCD_RS)
#define LCD_EN_OFF   PortEN&=~(1<<LCD_E)
#define LCD_D7_OFF   PortD7&=~(1<<LCD_D7)
#define LCD_D6_OFF   PortD6&=~(1<<LCD_D6)
#define LCD_D5_OFF   PortD5&=~(1<<LCD_D5)
#define LCD_D4_OFF   PortD4&=~(1<<LCD_D4)
#define BACKLIGH_OFF Puerto_BACKLIGH|=(1<<BACKLIGH)
#define BACKLIGH_ON Puerto_BACKLIGH&=~(1<<BACKLIGH)
#define              DISP_ON                    0b00001100
#define              DISP_OFF            0b00001000
#define              CLR_DISP            0b00000001
#define              CUR_HOME            0b00000010
#define              ENTRY_INC           0b00000110
#define              DD_RAM_ADDR         0b10000000
#define              DD_RAM_ADDR2  0b11000000
#define              CG_RAM_ADDR         0b01000000
#define              DISP_ON_CURSOR                     0b00001110
#define              DISP_ON_CURSOR_BLINK0b00001111
#define              ENTRY_INC_SHIFT                    0b00000111
#define       MOVE_CURSOR_RIGHT          0b00010100
#define       MOVE_CURSOR_LEFT           0b00010000
#define              SHIFT_RIGHT                        0b00011100
#define              SHIFT_LEFT                         0b00011000
void LCD_Inicia(void);
void LCD_Write(char caracter);
void LCD_Send_Command(int comando);
void LCD_Goto_XY( unsigned char X, unsigned char Y );
void LCD_String(char* data, int NBytes);
void LCD_Back_Ligh_On(void);
void LCD_Back_Ligh_Off(void);
void LCD_New_Caracter(unsigned int ubicacion, char* caracter);
#endif
```

ANEXO 3.- DISEÑO DE PISTAS Y PLACA

Parte inferior de las pistas en placa maestra::



Parte superior de las pistas en placa maestra



Placa maestra componentes: