Proyecto N° 1
Tarjeta de desarrollo ARDUINO

Daniela Matallana Caballero, Mateo Tafur Herrera, Luis Armando Cadena Ceballos.
Marzo 21 de 2017

I. RESUMEN:

Arduino es una plataforma electrónica de código abierto basado en hardware y software fáciles de usar. Con el fin de tener una introducción a los microprocesadores, se propone como primer proyecto la creación de una placa de Arduino teniendo en cuenta el funcionamiento del microcontrolador y el microprocesador como elementos principales de la placa. Después de plantear la idea, se desarrolla un diseño en el cual cada grupo de trabajo decide la forma, tamaño y distribución de su placa para una posterior implementación. En simultánea con el desarrollo del hardware de la placa, se estudia el funcionamiento del microcontrolador y se desarrollan diferentes actividades que permiten comprender de manera más práctica la importancia y funcionamiento de estos elementos. A continuación se describe todo el proceso llevado a cabo para la creación de la tarjeta de desarrollo MALUDA, la cual está basada en un microcontrolador con CPU AVR de 8 bits.

II. INTRODUCCION

"Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes interconectadas dentro del microcontrolador, y en conjunto forman lo que se conoce como microcomputadora. El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario le escribe. Se emplea para controlar el funcionamiento de una tarea determinada, y debido a su reducido tamaño, suele ser incorporado en el propio dispositivo al que gobierna". [1]

"El microcontrolador (MCU) ha sido la columna vertebral de los sistemas integrados más tiempo del que cualquiera tenga memoria. Los MCU son el procesador preferido para muchas aplicaciones en un sinfín de mercados. Los productos industriales y de consumo dependen de los microcontroladores. El pequeño tamaño del MCU, el bajo consumo de energía y las memorias compactas son la opción ideal para muchas aplicaciones. Los periféricos integrados y la memoria junto con los bajos precios son propuestas muy atractivas para los desarrolladores de sistemas". [2]

Como base fundamental para el desarrollo del proyecto, se requiere el uso de un microcontrolador con CPU AVR de 8 bits. Para ello, se hace necesario tener en cuenta las características de este tipo de microcontrolador.

"La arquitectura AVR es una de las arquitecturas líderes de 8 bits. Se ha convertido en sinónimo de facilidad de uso desde sus comienzos a mediados de la década de los 90 por Alf-Egil Bogen y Vegard Wollan, dos estudiantes de la universidad de ciencia y tecnología de Noruega. El AVR es el dispositivo más atractivo para que los estudiantes y aficionados conozcan más acerca de las arquitecturas informáticas y la programación integrada. El núcleo del procesador AVR fue uno de los primeros MCU en usar memoria flash en chip para el almacenamiento de memoria de los programas. Además, la arquitectura AVR permite la reprogramación en el sistema.

El núcleo de CPU de AVR es un motor RISC (computadora de conjunto de instrucciones reducido). RISC significa una complejidad reducida de instrucciones. Este conjunto de instrucciones enriquecido tiene ventajas para un compilador porque puede encontrar la instrucción de ciclo único óptima para su necesidad. El código de máquina resultante aumenta la velocidad y disminuye el uso de memoria en SRAM y Flash. La arquitectura AVR combina este conjunto de instrucciones enriquecido con 32 registros de trabajo de propósito general de 8 bits directamente conectados a la ALU (unidad aritmética lógica). La figura 1 muestra el funcionamiento interno de la arquitectura AVR. AVR usa la arquitectura Harvard, que separa memorias y buses de datos. Tiene una segmentación de nivel único de dos etapas. La mayoría de las instrucciones se ejecutan en un solo ciclo de reloj. Debido a esta ejecución de ciclo único de operaciones aritméticas y lógicas, los MCU basados en AVR entregan cerca de 1 MIPS (millones de instrucciones por segundo) por mega Hertz de frecuencia de reloj." [2]

Con un poco más de claridad acerca de la arquitectura

AVR se procede con el desarrollo del proyecto, desde el diseño del hardware y software hasta el resultado final.

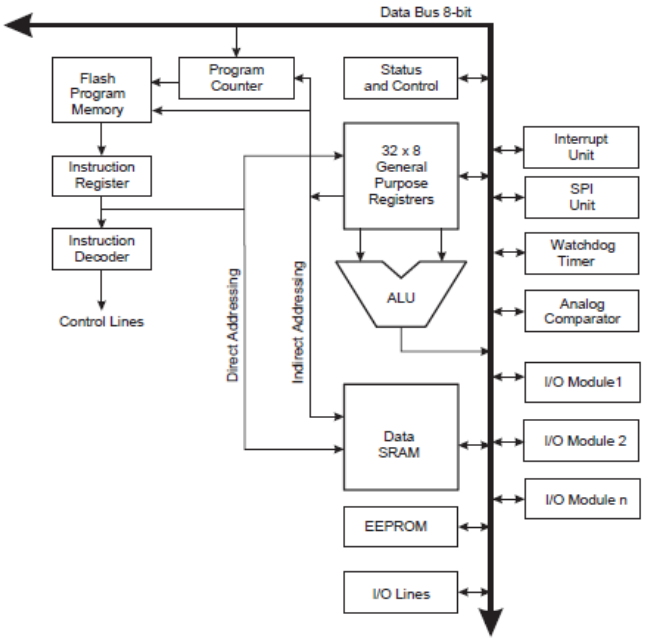

Fig.1 Arquitectura AVR. Tomado de [2].

III. MÉTODOS E INSTRUMENTOS.

Para el desarrollo de este proyecto, fueron necesarios los siguientes materiales y dispositivos electrónicos.

- Baquelita
- Diodo 1N4148
- Capacitores
- Led
- Resistencias
- Cable UTP
- Porta integrado
- Conectores hembra
- Regleta macho
- Microcontrolador ATMEGA 328
- Pulsador
- Oscilador de cristal 16 MHz
- FTDI232
- Cautín
- Estaño
- Tarjeta Arduino
- IDE Arduino
- Cable mini USB

Teniendo en cuenta que la mayoría de los elementos utilizados son conocidos, se realizará una breve explicación de aquellos elementos que no son tan comunes.

➢ MICROCONTROLADOR ATMEGA 328:

"El ATMEGA 328 AVR 8 bit es un circuito integrado de alto rendimiento que está basado en un microcontrolador RISC, combinando 32 KB ISP flash, una memoria con la capacidad de leer mientas escribe, 1 KB de memoria EEPROM, 2 KB de SRAM, 23 líneas de entrada/salida de propósito general, 32 registros de proceso general, 3 temporizadores flexibles/contadores con modo de comparación, interrupciones internas y externas, programador de modo USART, una interfaz serial orientada a byte de 2 cables, SPI puerto serial" [3], entre muchas otras cosas. La figura 2 permite observar la apariencia física de este integrado.

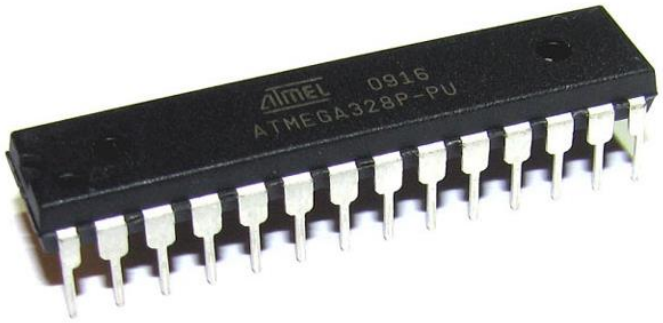

Fig.2 ATMEGA 328. Tomado de [3].

➢ Oscilador de cristal 16 MHZ:

"Un oscilador de cristal es un oscilador electrónico que utiliza la resonancia mecánica de un cristal vibratorio de material piezoeléctrico (la piezoelectricidad es un fenómeno que ocurre en determinados cristales que, al ser sometidos a tensiones mecánicas, en su masa adquiere una polarización eléctrica y aparece una diferencia de potencial y cargas eléctricas en su superficie) para crear una señal eléctrica con una frecuencia precisa. Esta frecuencia se utiliza comúnmente para controlar el tiempo, como en los relojes de cuarzo, para proporcionar una señal de reloj estable en circuitos integrados digitales y para estabilizar las frecuencias de los transmisores y receptores de radio". En la figura 3, se puede observar un oscilador de cristal de 16 MHz. [4]

Fig.3 Oscilador de cristal. Tomado de [4].

- **FTDI 232:**

La figura 4 presenta un FTDI 232. "El FTDI 232 es una interfaz USB a UART serie con la salida del generador de reloj opcional, y la nueva característica de seguridad dongle FTDI chip-ID. Además, los modos de interfaz poco explosión síncronas y asíncronas están disponibles. El FT232 añade dos nuevas funciones en comparación con sus predecesores. El reloj generado internamente puede ser llevado fuera del dispositivo y se utiliza para conducir un microcontrolador a lógica externa". [5]

Fig.4 FTDI 232. Tomado de [4].

- **Tarjeta Arduino:**

"La tarjeta Arduino es una placa que contiene un entorno de desarrollo, un microcontrolador, puertos de entrada y salida tanto digitales como análogos e interfaces de comunicación. Este dispositivo electrónico tiene como objetivo facilitar la realización de diferentes prototipos y múltiples diseños electrónicos, ya que este hardware, perteneciente a la plataforma Arduino, es de tipo libre y se puede utilizar sin necesidad de adquirir o comprar una licencia como ocurre con otras plataformas que contienen entorno de desarrollo. Arduino simplifica el proceso del trabajo con microcontroladores basados en ATMEGA8 y ATMEGA168 de Atmel." [6]. En la figura 5 se puede observar una placa de Arduino MEGA.

Fig.5 Arduino MEGA. Tomado de [6].

- **IDE Arduino:**

En la figura 6 se presenta la interfaz del IDE. "Dado que el Arduino es como un pequeño ordenador que ejecuta una serie de códigos que previamente le hemos introducido, necesitamos un programa para poder meter estos códigos a la propia placa. Este programa se llama IDE, que significa "Integrated Development Environment" ("Entorno de Desarrollo Integrado")". [7]

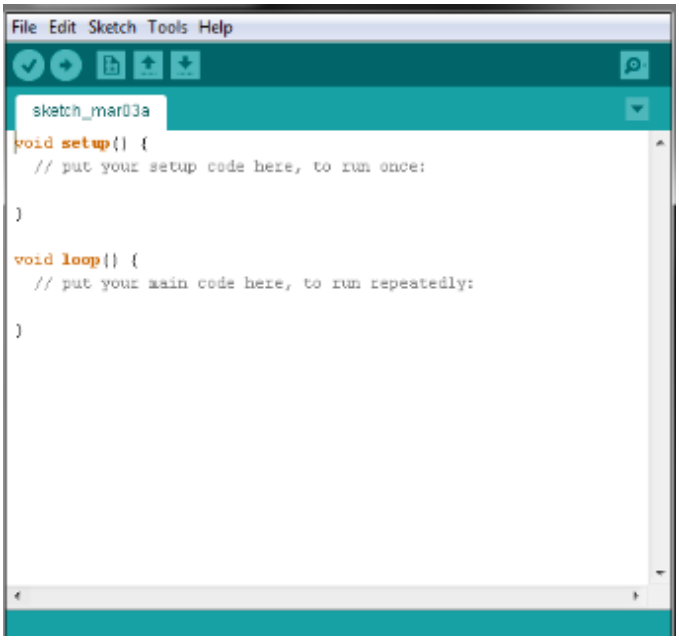

Fig.6 IDE Arduino. Tomado de [7].

IV. RESULTADOS Y DISCUSIÓN

A continuación se presenta una recopilación del proceso que se llevó a cabo para el desarrollo de la tarjeta de desarrollo MALUDA. El proyecto llevo a cabo por medio de entregas parciales y tareas intermedias dando cumplimiento a los requerimientos establecidos por el profesor.

El desarrollo del proyecto se realizaría por medio del IDE de Arduino, y usando una tarjeta Arduino, por lo tanto, se requería tener un conocimiento previo de las tarjetas y sus componentes, para ello, se llevó a cabo el desarrollo de la tarea 1, la cual consistía en el análisis técnico de una tarjeta de desarrollo Arduino, se debían especificar sus componentes más importantes. La figura 7 permite observar el producto de esta tarea, para la cual se seleccionó un Arduino Pro.

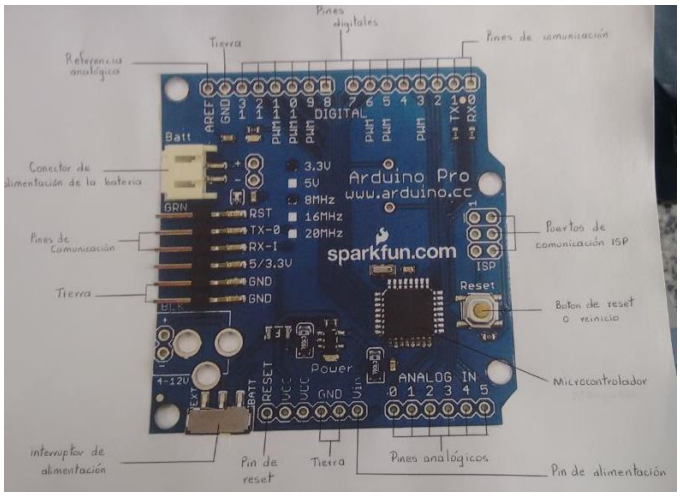

Fig.7 Revisión técnica del Arduino Pro.

Después de socializar la revisión de las tarjetas, era hora de empezar a crear un repositorio en el cual se pudieran tener todos los documentos del proyecto y todos los integrantes del grupo pudieran tener acceso a estos sin necesidad de estar en un mismo lugar. Este repositorio se creó por medio de GitHub, para lo cual se debería crear una cuenta y con comandos sencillos se lograban subir archivos desde el pc a la nube y compartirlos con los demás integrantes del grupo.

Luego, era necesario pensar en el diseño de la placa. Para esto, la creatividad jugo un papel importante, cada grupo realizó su diseño y un nombre representativo para su placa. La figura 8 permite observar el diseño de MALUDA, nombre que se le dio a la placa. Y continuando con el repositorio, se debía tener evidencia del trabajo, por esto, una vez aprobado el diseño por el profesor se debería cargar una imagen al repositorio. Es importante tener en cuenta que el profesor sugirió usar el FT232 para realizar la comunicación del microcontrolador de una manera más sencilla, puesto que hacerlo “a pedal” resultaba algo engorroso.

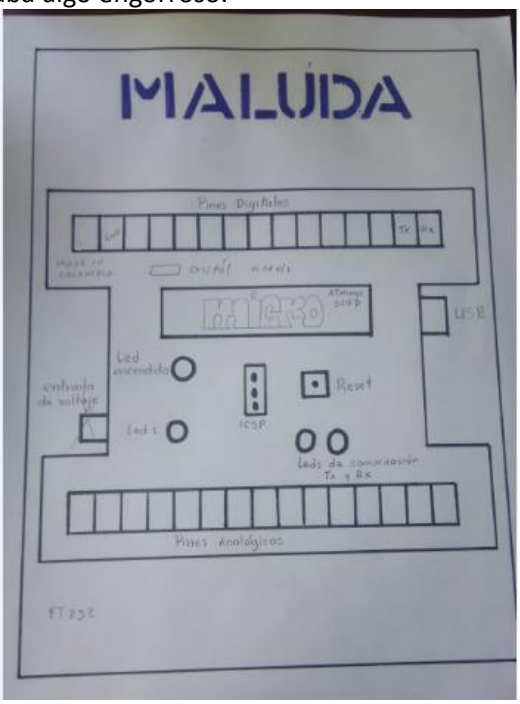

Fig.8 Diseño preliminar de MALUDA.

Después de tener el diseño listo, era necesario que la tarjeta se reconociera en el IDE de Arduino, para esto se creó un archivo .json, este archivo se subió a la nube al repositorio que se creó en GitHub. Allí, se obtuvo un enlace, el cual se debía copiar, luego en el IDE de Arduino, en preferencias se buscaba el gestor de URLS y allí se pegaba el enlace. Esto se hizo con el fin de que todos los cambios que se realizaran en la placa estuvieran disponibles como actualizaciones en el IDE de Arduino.

Una vez que se realizó este proceso, el nombre de la placa ya aparecía en el gestor de tarjetas del IDE de Arduino, solo faltaba instalarlo y ya aparecería como una opción en las placas del IDE de Arduino tal y como se muestra en la figura 9.

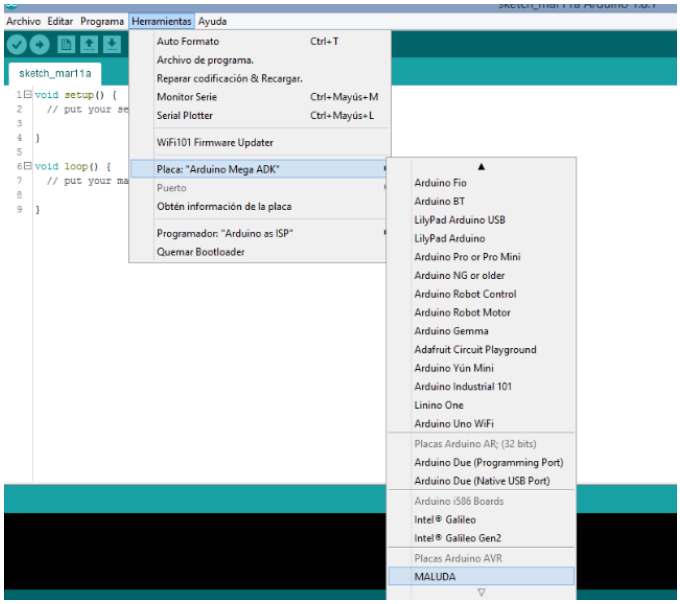

Fig.9 MALUDA en el IDE Arduino.

Una vez que se tenía disponible la opción de la tarjeta en el IDE, se crearía una versión de la tarjeta en el repositorio de GitHub para esto, fue necesario realizar un proceso en el cual se debía buscar la carpeta “avr” que se encuentra en la carpeta “archivos de programa (x86)” en el disco local C (si el IDE de Arduino fue instalado en este disco), allí en la carpeta “Arduino”, luego en “hardware” y por último “Arduino”. Una vez ubicada esta carpeta, se debería copiar y pegar en la carpeta del repositorio. Después, en la carpeta “bootloaders” se borran todos los archivos excepto la carpeta “optiboot”. Esta carpeta se conserva debido a que es la carpeta que contiene los archivos del Arduino UNO (el cual está sirviendo de base para el desarrollo de la tarjeta), las demás carpetas contienen los archivos de las demás tarjetas de Arduino, las cuales no son necesarias para este proceso. La carpeta “bootloaders” ya se deja quieta y regresaba a “avr”, allí se borra la carpeta “firmwares” puesto que contiene archivos innecesarios. Seguido de esto, por medio de atom se abre el archivo “Boards.txt”. Es importante tener en cuenta que este archivo a pesar de tener extensión .txt no se debe abrir con el bloc de notas porque sería imposible comprenderlo. En este archivo están todas las configuraciones para todas las tarjetas de Arduino, como solo se necesita la del Arduino UNO, las demás configuraciones se borran, y teniendo en cuenta que se están creando los archivos para la tarjeta que se creó, entonces el nombre UNO se debe cambiar por el nombre de la tarjeta, en este caso, MALUDA. De la misma manera se debe cambiar el nombre de la tarjeta. Después, en la carpeta “variants” dejar solo la carpeta “standard”. Luego, teniendo en cuenta la hoja de datos del microcontrolador, se hace un mapeo con el archivo pins_arduino.h; allí se pueden verificar las posiciones de los pines. En este archivo es posible cambiar los pines de posición, pero es importante recordar que no es recomendable cambiar de posición lo pines A0 y A1. Se guardan los cambios (si se realizaron cambios) y se sube la carpeta “avr” al repositorio. Con el comando “git checkout – b” se crea una versión. Para que las diferentes versiones aparezcan como actualizaciones en le IDE de Arduino, se requiere de otros datos como el tamaño del archivo de la versión (que se descarga desde el repositorio) entre otros. La versión debe aparecer en el repositorio tal y como se puede observar en la figura 10.

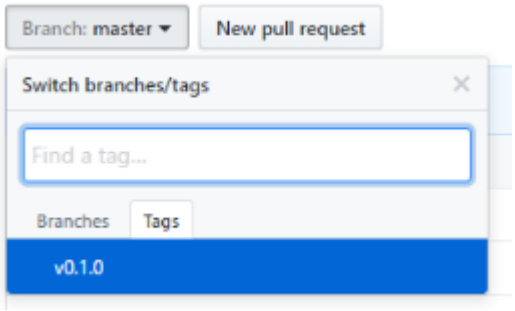

Fig.10 Versión 0.1.0 de MALUDA.

Como ya se tenía una versión de la placa, era necesario conocer el funcionamiento interno del microprocesador, y para ello, la segunda tarea, la cual consistía en realizar la explicación de un procesador (CPU) AVR de 8 bits usando una maqueta que describa el funcionamiento de cada uno de los elementos que la componen y la conexión entre ellos. Con el fin de hacer algo más didáctico y fácil de entender, se realiza una maqueta con imágenes alusivas a los componentes del micro. La figura 11 ilustra la maqueta. Esta maqueta se realizó teniendo en cuenta la información de la hoja de datos del integrado y la información que se obtuvo de internet.

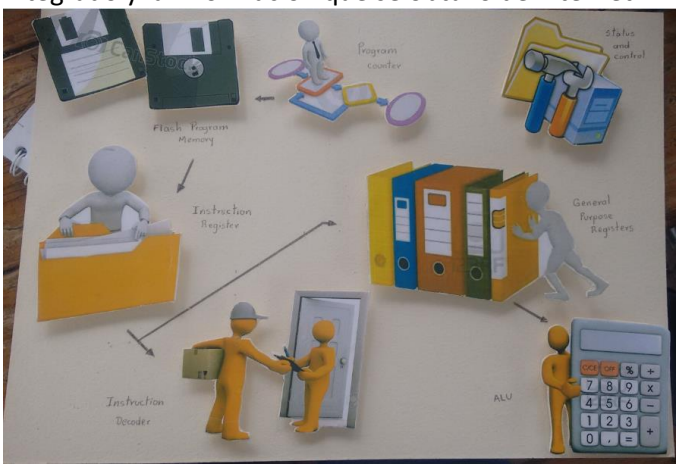

Fig.11 Maqueta procesador AVR de 8 bits.

Una vez comprendido el funcionamiento del procesador, el segundo avance, el cual consistía en el desarrollo del PCB siguiendo el diseño propuesto inicialmente. En la figura 12 se puede observar el diseño final del PCB puesto que fue necesario modificar varias veces este documento puesto que se encontraban errores que debían ser corregidos. El más significativo de ellos fue la distancia del cristal al micro, pues inicialmente estaba un poco retirado, y, con solo mover este componente el diseño cambio en gran medida. La evidencia de este avance se debía subir al repositorio.

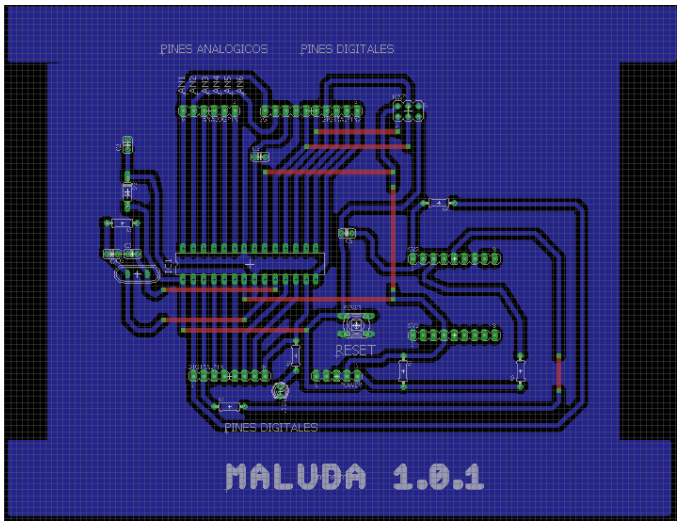

Fig.12 Diseño PCB.

Finalmente, después de corregir todos los errores y mejorar un poco el diseño, el tercer avance, que correspondía al circuito impreso. En la figura 13 se puede observar como quedo el circuito quemado en la baquela. La evidencia es este avance también debería aparecer en el repositorio.

Fig.13 Circuito impreso.

Para comprobar que el circuito estuviera funcionando, se conectó al pc y se esperaba que el Led de la placa encendiera. Para esto, surgieron varios inconvenientes; el primero fue que no había continuidad en los caminos del cristal y por lo tanto pues el Led nunca iba a encender. Con un poco de estaño se solucionó este inconveniente. Luego, el Led debería ir al pin 13, pero quedo en el pin 7. Por lo tanto, fue necesario recurrir al archivo de "pins_arduino.h" para realizar el mapeo y cambiar los pines de posición. Sin embargo, el Led seguía sin encender. La solución fue quitar unas resistencias que estaban conectadas al FT232, puesto que este aparato ya trae todo incorporado. Una vez que se realizaron todos estos cambios, la tarjeta funcionó de manera adecuada. Ahora, solo faltaba quemar el bootloader. Para esto, se tenía una guía compartida por el profesor, en donde se trabajaba con una tarjeta Arduino UNO como programador y se mostraban las conexiones necesarias. En el grupo se cuenta con un Arduino MEGA, por lo que fue necesario buscar la ubicación de los pines MISO, MOSI, SCK y SS en este Arduino. Una vez se supo la ubicación de estos pines se quemó el bootloader en la placa MALUDA y se comprobó que funcionara de manera adecuada. Para evitar posibles confusiones con la numeración de los pines, se marcaron todos con su respectivo nombre así como se muestra en la figura 14.

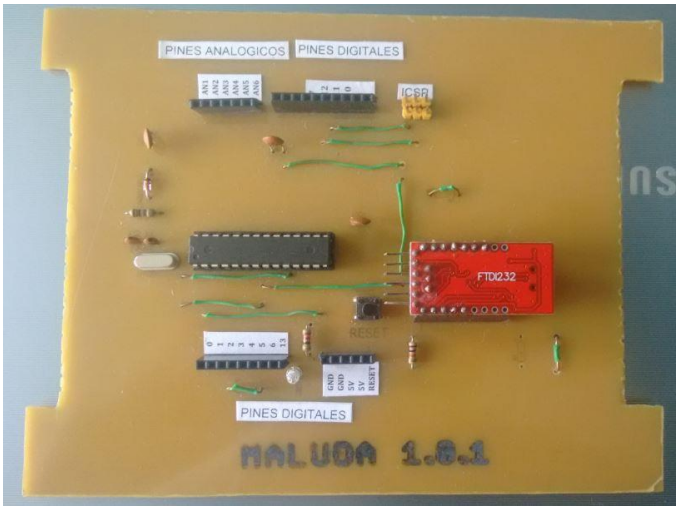

Fig.14 Tarjeta de desarrollo MALUDA.

V. ANÁLISIS DE RESULTADOS

- En base al esquemático del ARDUINO (puesto que este maneja una licencia libre para la interactividad con los desarrolladores) y teniendo en cuenta el datasheet del

microprocesador (ATMEGA328) y con la ayuda del software de simulación EAGLE se creó un esquemático basado en el funcionamiento del ARDUINO UNO imitándolo en algunas funciones.

- En el diseño de la placa se utilizó un conversor USB a serial FT232 ya que su función es la transmisión de datos de la tarjeta de desarrollo como también la función de alimentar a esta; fue necesario utilizarla ya que las implementaciones de sus funciones representan complejidad.
- Cabe resalta que para la comunicación entre la FT232 con el microprocesador (ATMEGA328) hay que tener en cuenta que la conexión debe ser de RX a TX y TX a RX puesto que RX (recepción, bajar, descargar) como TX (transmisión, subir) cumplen con funciones opuestas.
- Como la FT232 viene con una resistencia desde su circuito eléctrico no se deben tener resistencias en la comunicación con el microprocesador (ATMEGA328) puesto que éstas causarían la no transmisión de datos es, decir, la comunicación entre estos.
- De acuerdo a las funciones del software de simulación se tuvo muy en cuenta la posición del cristal con respecto al microprocesador puesto que este al estar muy alejados presentan una variación en la frecuencia que afectaría el correcto desarrollo.
- En la creación del esquemático se tienen algunas especificaciones algunas de ellas como: el ancho de línea que conecta los componentes (1.016mm), también se tiene en cuenta la curvatura de línea ya que no es muy aconsejable tenerla con ángulo de 90° puesto que la corriente no tendría un buen tránsito.
- Al momento de poner el Led en el pin 13 del microcontrolador, el Led quedo el pin físico, pero internamente quedo en el pin 7. Por esto, se debió cambiar la posición de estos pines por medio de un mapeo en el archivo pines.h.
- Se debe cargar el BOOTLOADER el cual es necesario para el almacenamiento del programa que actualiza la aplicación. Esto se conoce como Auto-programación y es una característica que permite actualizar la aplicación sin necesidad de programadores externos.
- Para quemar el bootloader en la tarjeta que se creó, se necesario tener otra tarjeta que va hacer la función de programador. Para esto, es necesario conectar los pines MOSI, MISO, SCK y SS, y aplicar los comandos Tools -> Board, Tools programmer y Tools -> Burn bootloader.

VI. CONCLUSIONES

- Se logró desarrollar una plataforma con prototipos similares a los de ARDUINO para el desarrollo de sistemas embebidos en proyectos multidisciplinarios con flexibilidad y sencillez a la hora de su utilización.
- Se logró entender el funcionamiento teniendo en cuenta el marco teórico para el desarrollo de la tarjeta se tiene la idea de cómo se ejecuta una instrucción a lo largo de los dispositivos al llegar la instrucción a la memoria flash luego el trabajo que hace el estado y control donde se encuentran el contador de programa que es el encargado de contener la dirección donde se almacena dicha instrucción y al mismo tiempo el registro de instrucción el cual contiene toda la información lista para ser codificada, de este modo el decodificador de instrucciones es el encargado de descifrar la clase de datos que contiene el registro de instrucciones y por ende el decodificador es el encargado de enviar ya en el lenguaje indicado para que esta sea ejecutada y procesada respectivamente.
- Se pudo evidenciar la flexibilidad y lo grato que es trabajar con placas ARDUINO puesto que contienen tanto software como hardware. Los entornos de desarrollo y lenguaje de programación de ARDUINO y las placas en las que se ejecutan son desarrollados de la mano, por lo que tenemos asegurada tanto la compatibilidad como la sencillez de desarrollo sobre ellas. A diferencia de otras placas y microcontroladores no se realiza de esta manera.
- Se comprobó la utilización para el desarrollo de múltiples funciones como elementos autónomos, controlar un elemento, transformar la información de una fuente, o bien conectarse

a otros dispositivos o interactuar con otros programas, para interactuar tanto con el hardware como con el software.

VII. REFERENCIAS

[1] Ingeniería electrónica y proyectos. Electrónica estudios. “Qué es un microcontrolador” [online] disponible en: http://www.electronicaestudio.com/microcontrolador.htm

[2] Microcontroladores AVR para procesamiento de 8 bits de alto rendimiento y consumo eficiente de energía [online] disponible en: https://www.arrow.com/es-mx/research-and-events/articles/avr-microcontrollers-for-high-performance-and-power-efficient-8-bit-processing

[3] Microcontrolador ATMEGA 328 especificaciones técnicas [online] disponible en: https://es.wikipedia.org/wiki/Atmega328

[4] Oscilador de cristal de 16MHz [online] disponible en: https://es.wikipedia.org/wiki/Oscilador_de_cristal

[5] Future technology devices international Ltd. The USB bridging solutions specialist “FT232R – USB UART IC” [online] disponible en: http://www.ftdichip.com/Products/ICs/FT232R.htm

[6] What is Arduino? [Online] disponible en: www.arduino.cc

[7] Martínez F. “Tutorial Arduino: IDE Arduino” [online] disponible en: https://openwebinars.net/blog/tutorial-arduino-ide-arduino/