

UNIVERSIDAD TECNOLÓGICA
NACIONAL

FACULTAD REGIONAL CÓRDOBA

INGENIERÍA ELECTRÓNICA

“Proyecto Integrador de 4to año:
ARMando”

Integrantes:
CHIOSSO, Ezequías
CONDORÍ, Emanuel David
ZALAZAR, Facundo Emmanuel

Curso 4R1

13 de febrero de 2015

Índice general

1. Objetivos	2
1.1. Objetivos complementarios	2
2. Introducción	3
3. Principio de Funcionamiento.	5
3.1. Procesador ARM.	5
3.2. Interacción con el usuario.	5
3.2.1. Vía Bluetooth	5
3.2.2. Display LCD en “ARMando”	6
3.3. Movilidad y control.	6
3.4. Escaneo	9
4. Análisis técnico.	10
4.1. Placa ARM	10
4.2. Modulo Bluetooth HC06 o Linvor	10
4.2.1. Pines	11
4.2.2. Esquema de conexión	12
4.2.3. Código necesario	13
4.2.4. Comunicación con celular	14
4.3. Sensor CNY70 para Seguidor de linea	14
4.4. Sensor detector de movimiento.	15
4.5. Sensor detector de fuego	16
4.6. Sensor de temperatura LM35	17
4.7. Puente H	18
4.7.1. Funcionamiento del Puente H	19
4.8. Pantalla LCD 1602A	20
4.8.1. Parámetros Máximos Absolutos de Funcionamiento	20
4.8.2. Características Eléctricas	21
4.8.3. Código	21
5. Circuito General	23
6. Programación	25
6.1. Código	25
6.2. Análisis	31
7. Conclusión	32

Capítulo 1

Objetivos

Es objetivo primordial de este proyecto integrador poder aplicar y afianzar los conocimientos incorporados en las distintas materias a lo largo de este año.

Para cumplir con lo dicho es que se ha optado por crear y diseñar un dispositivo que permita unir algunos de estos conceptos aprendidos en un solo proyecto.

Es así como nace el prototipo de auto de seguridad: “ARMando”.

1.1. Objetivos complementarios

Dentro del objetivo principal se encuentran entrelazados otros objetivos que permiten cumplir con el primordial. Entre ellos:

- Aprender a utilizar el procesador ARM.
- Utilizar distintos dispositivos externos.
- Programar una placa, lo cual permite que los distintos dispositivos cumplan con la función deseada.
- Cálculos y mediciones de distintos parámetros para el buen funcionamiento del circuito.
- Circuitos acopladores de señal (CAS), los cuales transforman los rangos de señales para luego ser ingresados a la placa central.
- Aprender a utilizar las protecciones necesarias para posibles accidentes.
- Control de motores.
- Interacción con el usuario y control mediante Bluetooth.

Capítulo 2

Introducción

El prototipo de seguridad “ARMando” es un vehículo de tamaño reducido, que tiene como función escanear y analizar un área deseada, y avisar al usuario ante cualquier desperfecto.

Su nombre se debe a que el centro que lo comanda es una placa con procesador ARM. Y de ésta depende el funcionamiento y la unificación de las partes.

Posee dos modos de uso: automático o manual. El funcionamiento de ambos se detallará más adelante, pero es el usuario el que elige la opción mediante una terminal de Bluetooth.

Sea cualesquiera de ambos modos, el vehículo se desplaza y escanea en los lugares que uno desee.

Los parámetros del ambiente a revisar son: fuego, movimiento y temperatura del lugar. (Este es un prototipo, y por lo tanto si se desea, se pueden ampliar los rangos a analizar).

Luego de escaneada el área, “ARMando” envía los datos recogidos a la terminal Bluetooth usada.

(Luego de receptada la información se podría crear un sistema que avise vía web lo sucedido, o mediante mensaje de texto. En este prototipo no fue hecho)

A lo largo del informe se irá explicando de forma más precisa cómo funciona cada parte, y cómo se unifica con el resto.

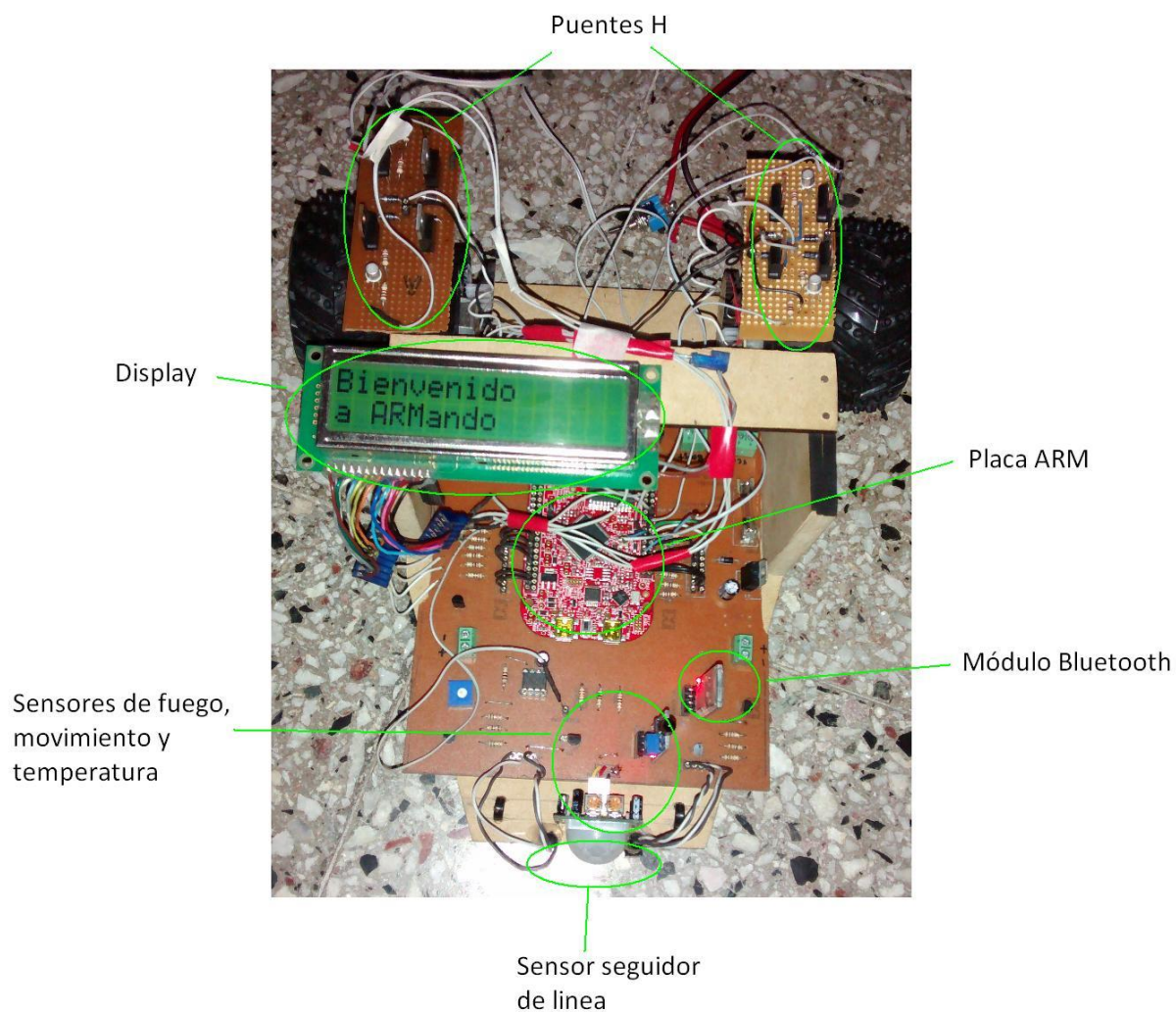


Figura 2.1: Partes de ARMando

Capítulo 3

Principio de Funcionamiento.

Para desarrollar mejor este tema y poder hacerlo comprensible al lector, es que se optó por separarlo en sus distintas partes:

3.1. Procesador ARM.

Como ya se dijo, este es el “cerebro” que comanda el vehículo. Lo hace mediante las instrucciones provistas por el código desarrollado (El programa completo se encuentra en los siguientes capítulos). De esta forma el procesador con su código cargado, es el aglutinante y unificador de las distintas partes. Gracias a éste es que se puede tomar y analizar la información provista por los sensores, y tomar una decisión según las circunstancias.

También mediante el procesador es que uno puede interactuar con el vehículo a través de los módulos Bluetooth.

En fin el procesador ARM, como lo dice su nombre, es el encargado de recibir información, *procesarla* mediante el código, y producir una acción deseada.

3.2. Interacción con el usuario.

3.2.1. Vía Bluetooth

La persona que utiliza “ARmando” se comunica con éste vía Bluetooth. Esto puede hacerse mediante un celular o una PC que posea este servicio.

Una vez que se enciende el vehículo de forma manual, el procesador activa el módulo Bluetooth. En el receptor aparece con el nombre de “linvor” y la contraseña por defecto es 1234.

Una vez enlazados ambos dispositivos aparece en la pantalla las siguientes opciones:

1. Modo automático
2. Modo manual
3. Ver datos

Modo automático

El vehículo sigue el recorrido trazado (el cual es una franja negra con bordes blancos) utilizando los *sensores seguidores de línea* colocados al frente. En la siguiente sección se profundiza sobre cómo se logra esto, pero por ahora es útil saber lo básico.

Una vez que comienza a moverse está programado para que cada cierta distancia se frene, escanee y vuelva a arrancar.

El final del recorrido se marca con una franja blanca en la pista, la cual es detectada por los sensores. En ese punto “ARmando” envía los datos registrados al celular vía Bluetooth, y vuelve a comenzar.

Modo manual

En este caso, el vehículo se desplaza a discreción del usuario. Ya no utiliza los sensores para guiarse, sino que se maneja mediante las teclas A(izquierda) S(atrás) D(derecha) W(adelante) Q(frenado). Como se puede notar en esta función existe la posibilidad de ir hacia atrás.

En cualquier momento el usuario puede activar el escaneo del área, y luego cuando se desee se envían los datos al celular.

Ver Datos

Esta opción permite ver todos los datos recogidos hasta el momento. Una vez apagado la conexión estos son borrados.

3.2.2. Display LCD en “ARMando”

“ARMando” posee adosado un display que muestra, en tiempo real, los datos recogidos y las opciones elegidas por el usuario. Esto es de utilidad a la hora de verificar si el funcionamiento de las conexiones y del escaneo son correctos.

3.3. Movilidad y control.

Para la mecánica se utilizaron dos motores de corriente continua. Cada uno va colocado en una rueda trasera, y se encargan no sólo de la tracción, sino también del giro.

Cuando ambas ruedas se activan el vehículo se desplaza. Pero cuando se realiza un giro cada rueda tracciona en sentido opuesto, de modo que la rotación es instantánea y en el lugar. Este tipo de rotación se logró mediante la utilización de *Puentes H*, los cuales permiten la inversión de polaridad en los motores. En la sección de Análisis Técnico se explica de forma detalla su funcionamiento, pero a continuación se explicará los tipos de respuesta que produce para poder entender el funcionamiento del vehículo.

Cada *Puente H* posee 2 entradas, y a la salida está conectado el motor. De esta forma, cada Puente H responde a la siguiente tabla de verdad:

Entada 1	Entrada 2	Movimiento del motor
1	1	Frenado
1	0	Avance
0	1	Retroceso
0	0	Frenado

Cuadro 3.1: Lógica del Puente H

En el **modo manual** el movimiento depende del comando del usuario:

Comando	Puente H Derecho		Motor Derecho	Puente H Izquierdo		Motor Izquierdo	Resultado
W	1	0	Avanza	1	0	Avanza	Avance
	0	1		0	1		
D	1	0	Avanza	0	1	Retrocede	Giro Derecha
	0	1		1	0		
A	1	0	Avanza	0	1	Retrocede	Giro Izquierda
	0	1		1	0		
Q	0	0	Frenado	0	0	Frenado	Frenado
	0	1		0	1		
S	0	1	Retrocede	0	1	Retrocede	Retroceso

Cuadro 3.2: Lógica de manejo en modo manual

Como ya se dijo, en **modo automático**, el vehículo sigue el recorrido trazado utilizando los *sensores seguidores de línea*. Estos se utilizaron para detectar si lo que hay debajo es negro o blanco. En este caso se diseñaron de forma tal que si es negro envíe 1 y si es blanco envíe 0.

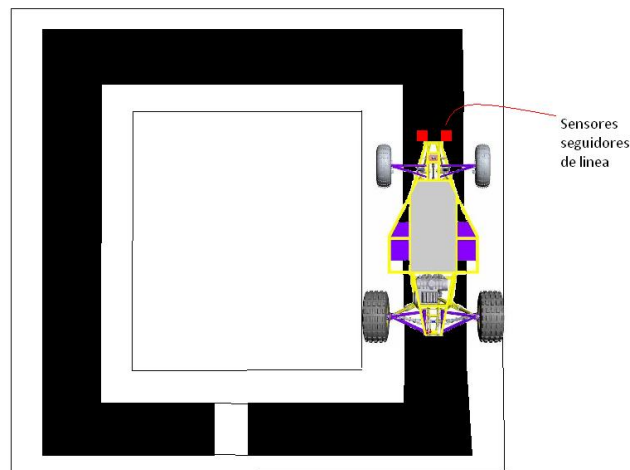


Figura 3.1: Modulo HC06

El comportamiento de los motores se muestra a continuación:

Sensor derecho	Sensor Izquierdo	Puente H Derecho	Motor Derecho	Puente H Izquierdo	Motor Izquierdo	Resultado
1	1	1 0	Avanza	1 0	Avanza	Avance
1	0	0 1	Retrocede	1 0	Avanza	Giro Derecha
0	1	1 0	Avanza	0 1	Retrocede	Giro Izquierda
0	0	0 0	Frenado	0 0	Frenado	Frenado

Cuadro 3.3: Lógica de manejo en modo automático

3.4. Escaneo

Existen 3 parámetros a censar:

1. **Movimiento:** el sensor encargado de esto posee un material especial que es sensible a las radiaciones infrarrojas. Cuando un cuerpo caliente como un humano o animal pasa frente al él, este lo detecta. El alcance es de 6 metros.
2. **Fuego:** este modulo es sensible a la radiación, y puede detectar fuentes de luz con una longitud de onda entre 760nm y 1100nm, por lo tanto la radiación perteneciente al fuego es detectada. La distancia máxima de detección es de 1 metro para una flama pequeña, por ejemplo de un encendedor. Pero si la fuente de radiación es mayor, lo que sucede en un incendio, se pueden detectar a distancias mayores.
3. **Temperatura:** este sensor mide la temperatura ambiente con una precisión calibrada de 1°C. Su rango de medición abarca desde -55°C hasta 150°C.

Como ya se estudió, en modo automático el escaneo del ambiente se realiza cada cierto período de tiempo según se desee. En el modo manual, este se realiza cuando se le envía la orden.

Sea cual sea el caso, para realizar el escaneo de una zona deseada, el vehículo debe estar en reposo, ya que de esta forma el sensor de movimiento puede efectuar bien las mediciones. En el caso contrario, el sensor no sabría diferenciar si el que se mueve es él o los objetos que lo rodean.

Capítulo 4

Análisis técnico.

4.1. Placa ARM

La placa utilizada es la Freescale KL46Z, con un procesador ARM Cortex M0+.

Esta placa cuenta con un acceso simple a los pines de entrada y salida de datos, es de bajo consumo y esta preparada para trabajar con pilas o baterías. Permite la expansión de la placa, e incluye una interfaz flash para la programación de la misma. En ella vienen incluidos un magnetómetro, sensor de luz visible, acelerómetro, un pequeño display LCD de 4 dígitos y un panel táctil capacitivo.

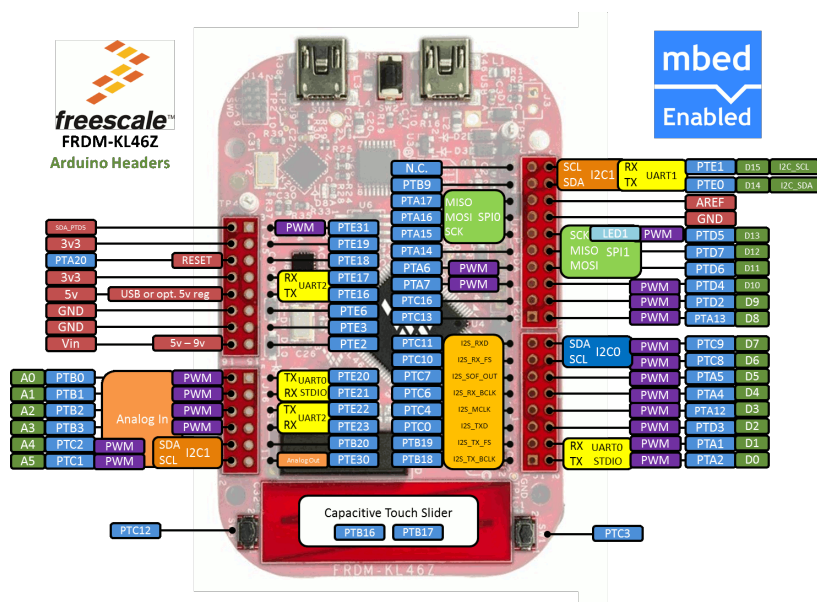


Figura 4.1: Freescale KL46z

4.2. Modulo Bluetooth HC06 o Linvor

Este modulo trabaja como maestro y esclavo, el nombre al buscarlo por Bluetooth es "Linvor", la contraseña es 1234, pero puede ser cambiada. Está configurado por defecto en 9600 baudios (lo cual también se puede modificar). Si el dispositivo no se encuentra enlazado el LED parpadea, en caso contrario se mantiene encendido, indicando la conexión.

El consumo durante el enlace varia en un rango de 30 a 40 mA. La media es de 25 mA. Luego de la comunicación la corriente se mantiene en 8 mA.



Figura 4.2: Modulo HC06

4.2.1. Pines

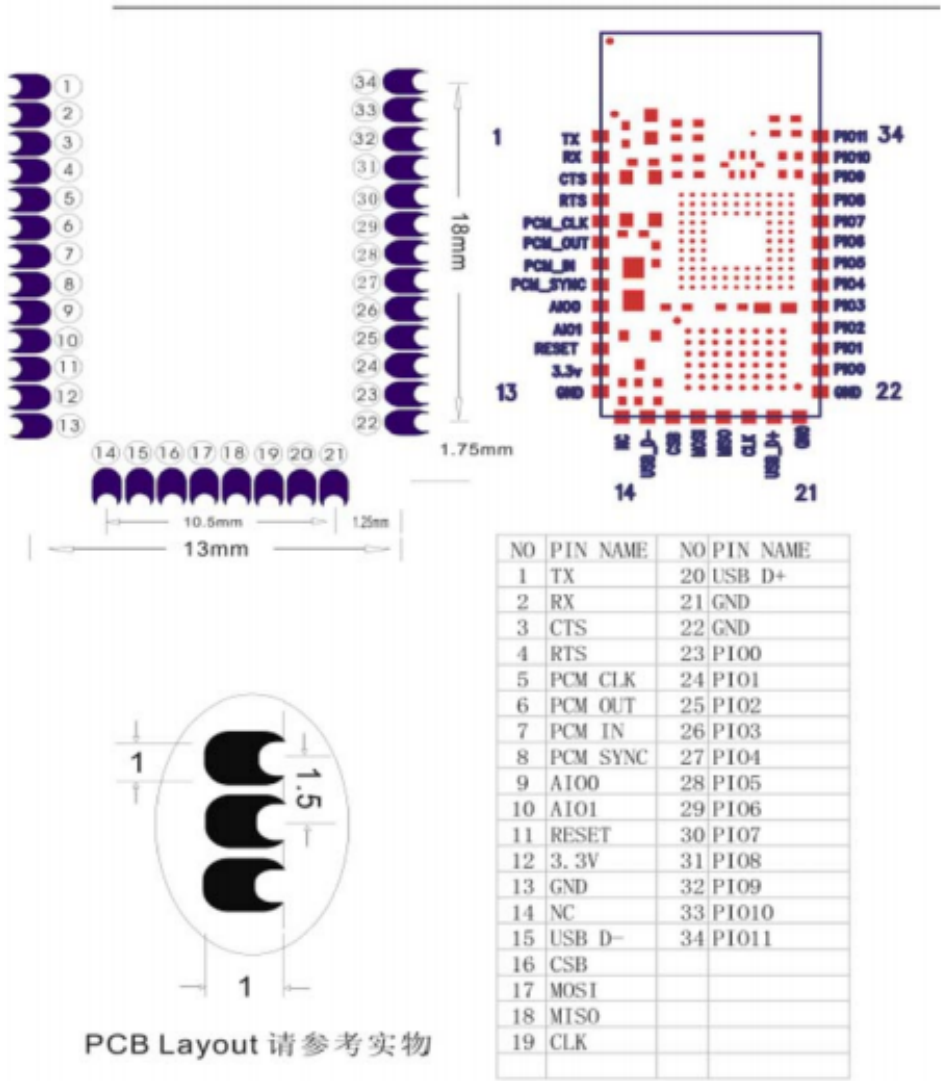


Figura 4.3: Esquema de los Pines

PIN	Descripción
PIN1	UART_TXD, nivel TTL/CMOS, datos de salida UART.
PIN2	UART_RXD, nivel TTL/CMOS, datos de entrada UART.
PIN11	Reset, el pin de reset del modulo.
PIN12	V_{CC} , tensión de alimentación el nivel estandar es 3,3 V, y puede trabajar entre 3 y 4,2 V.
PIN13-14	GND

Cuadro 4.1: Pines del Modulo HC06

4.2.2. Esquema de conexión

Linvor	KL46Z nombre de la señal	FRM-KL46Z
V_{CC}	$V_{DD}(3,3V)$	J3-4
GND	GND	J3-14
TXD	$PTE1/UART1_RX$	J2-20
RXD	$PTE0/UART1_TX$	J2-18

Cuadro 4.2: Pines

4.2.3. Código necesario

```
#include <RawSerial.h>
```

Inherits [mbed::SerialBase](#).

Public Member Functions

RawSerial (PinName tx, PinName rx)

Create a **RawSerial** port, connected to the specified transmit and receive pins.

```
int putc (int c)
```

Write a char to the serial port.

```
int getc ()
```

Read a char from the serial port.

```
int puts (const char *str)
```

Write a string to the serial port.

```
void baud (int baudrate)
```

Set the baud rate of the serial port.

```
void format(int bits=8, Parity parity=SerialBase::None, int stop_bits=1)
```

Set the transmission format used by the serial port.

```
int readable ()
```

Determine if there is a character available to read.

```
int writable ()
```

Determine if there is space available to write a character.

```
void attach (void(*fptr)(void), IrqType type=RxIrq)
```

Attach a function to call whenever a serial interrupt is generated.

```
template<typename T>
```

```
void attach (T *tptr, void(T::*mptr)(void), IrqType type=RxIrq)
```

Attach a member function to call whenever a serial interrupt is generated.

```
void send_break()
```

Generate a break condition on the serial line.

```
void set\_flow\_control (Flow type, PinName flow1=NC, PinName flow2=NC)
```

Set the flow control type on the serial port.

Figura 4.4: Librería RawSerial

```

1 #include "mbed.h"
2 RawSerial pc(PTE0, PTE1);
3 .
4 .
5 .
6 int main()
7 {
8 .
9 .
10
11     pc.putc(13); //Pone el cursor al comienzo de la pantalla en la aplicacion del celular
12     pc.printf("\n          ****Bienvenido****          \n"); //es enviado al modulo para que luego se visualice en el celular
13     do
14     {
15         pc.putc(13); //putc envia 13 en codigo ASCII
16         pc.printf("          1_Modo automatico          \n");
17         pc.putc(13);
18         pc.printf("          2_Modo Manual          \n");
19         pc.putc(13);
20         pc.printf("          3_Ver datos          \n");
21         pc.putc(13);
22         pc.printf("          4_Salir          \n");
23         elec=pc.getc(); //recibe el la opción ingresada por el usuario y lo guarda en elec
24         pc.putc(elec);
25         enter=pc.getc();
26     }while(S2<elec||enter!=127||elec<49);

```

Figura 4.5: Descripción de los comandos necesarios

4.2.4. Comunicación con celular

Para la comunicación entre modulo y el celular es necesario enviar los caracteres en codigo ASCII . Se ha usado la aplicación BlueTerm de Android que se puede descargar desde el Google Play.

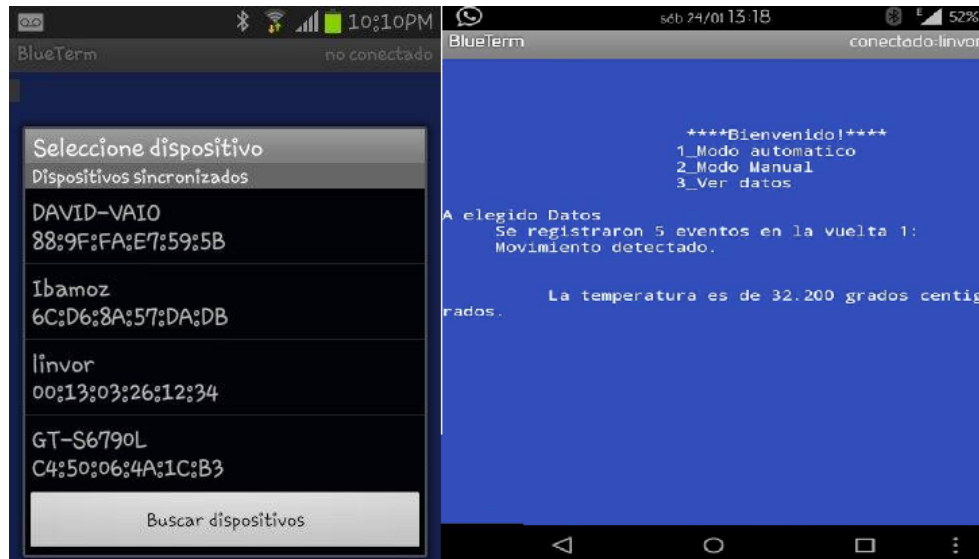


Figura 4.6: Imagen Ilustrativa BlueTerm

4.3. Sensor CNY70 para Seguidor de linea

El CNY70 es un sensor de infrarrojos de corto alcance basado en un emisor de luz y un receptor, ambos apuntando en la misma dirección, y cuyo funcionamiento radica en la capacidad de reflexión del objeto, y la detección del rayo reflejado por el receptor.

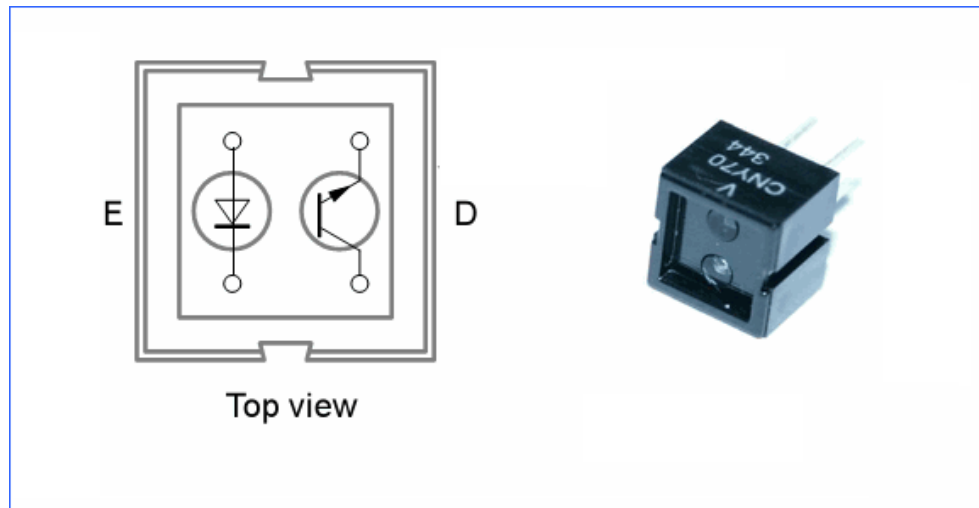


Figura 4.7: Sensor CNY70

Este sensor devuelve por la pata de salida correspondiente, según el montaje, un voltaje relacionado con la cantidad de rayo reflejado por el objeto. Para el montaje A, se leerá del emisor un '1' cuando se refleje luz y un '0' cuando no se refleje. Para el montaje B los valores se leen del colector, y son los contrarios al montaje A.

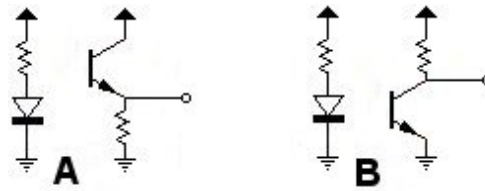


Figura 4.8: Tipos de configuraciones para el sensor

Si conectamos la salida a una entrada digital del microcontrolador, entonces obtendremos un '1' o un '0' en función del nivel al que el microcontrolador establece la distinción entre ambos niveles lógicos. Este sistema es el que se emplea para distinguir entre blanco y negro, en la conocida aplicación del robot seguidor de línea.

Otra posibilidad es conectar la salida a una entrada analógica. De este modo, mediante un conversor A/D se pueden obtener distintos valores. Esto permite la detección dinámica de blanco y negro (muy útil cuando el recorrido presenta alteraciones en la iluminación). Pero también, si se emplea el sensor con objetos de distintos color, se puede establecer un mecanismo para la detección de los distintos colores, determinando los valores marginales que separan unos colores de otros.

4.4. Sensor detector de movimiento.

Para comenzar a explicar cómo funciona un sensor básico, vamos a utilizar el diagrama que se muestra en la siguiente figura. El sensor PIR posee dos ranuras, cada una está hecha de un material especial que es sensible a las radiaciones infrarrojas. Cuando un cuerpo caliente como un humano o animal pasa frente al él, primero intercepta una mitad de el sensor PIR, lo que provoca un cambio diferencial positiva entre las dos mitades. cuando el cuerpo caliente sale de la zona de detección, que sucede a la inversa, por lo que el sensor genera una cambio diferencial negativo. Estos pulsos de cambio son los que se detectan y pueden utilizarse para saber si hay movimiento humano.

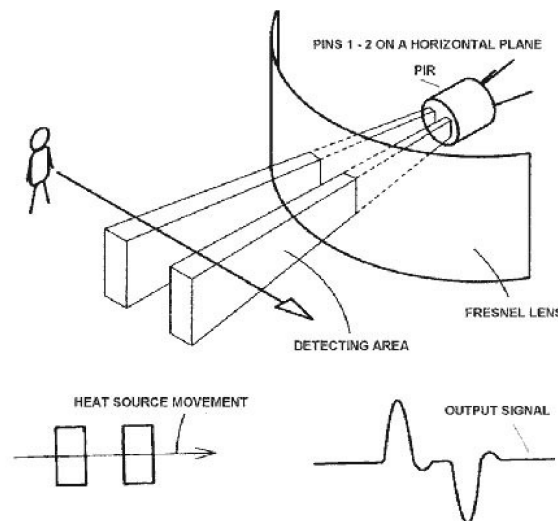


Figura 4.9: Principio de funcionamiento.

Además, para un mayor alcance (que es de 6 metros), se utiliza un lente de Fresnel.

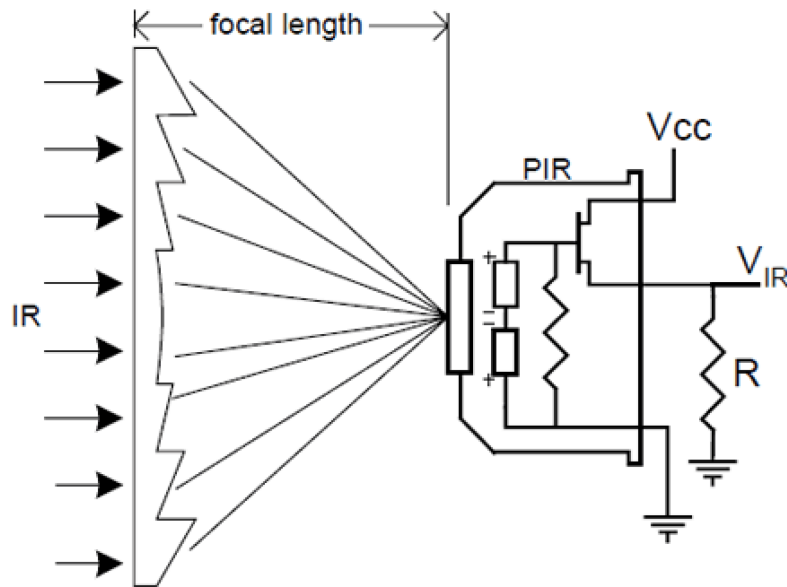


Figura 4.10: Lente de Fresnel en el sensor. Este condensa la luz y permite una detección a mayor alcance.

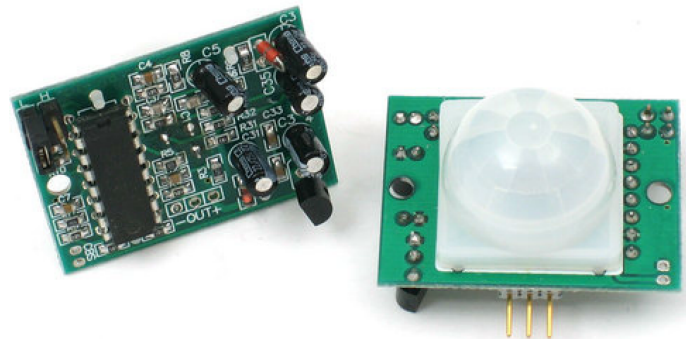


Figura 4.11: Sensor en detalle

Pin Definitions and Ratings

Pin	Name	Function
-	GND	Connects to Ground or Vss
+	V+	Connects to Vdd (3.3V to 5V) @ ~100uA
OUT	Output	Connects to an I/O pin set to INPUT mode (or transistor/MOSFET)

Figura 4.12: Pines

4.5. Sensor detector de fuego

Este modulo es sensible a la radiación, y puede detectar fuentes de luz con una longitud de onda entre 760nm y 1100nm, por lo tanto la radiación perteneciente al fuego es detectada. La distancia máxima de detección es de 1 metro para una flama pequeña, por ejemplo de un encendedor. Pero si la fuente de radiación es mayor, lo que sucede en un incendio, se pueden detectar distancias mayores.



Figura 4.13: Sensor de fuego.

Description

- Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm
- Detection distance: 20cm (4.8V) ~ 100cm (1V)
- Detection angle about 60 degrees, it is sensitive to the flame spectrum.
- Comparator chip LM393 makes module readings stable.
- Adjustable detection range.
- Operating voltage 3.3V-5V
- DO digital switch outputs (0 and 1)
- Power indicator and digital switch output indicator

Figura 4.14: Características del sensor de fuego.

4.6. Sensor de temperatura LM35

El LM35 es un sensor de temperatura con una precisión calibrada de 1°C . Su rango de medición abarca desde -55°C hasta 150°C .

Sus características más relevantes son:

- Está calibrado directamente en grados Celsius.
- La tensión de salida es proporcional a la temperatura.
- Tiene una precisión garantizada de 0.5°C a 25°C .
- Opera entre 4 y 30 voltios de alimentación.
- Baja impedancia de salida.
- Baja corriente de alimentación ($60\mu\text{A}$).

- Bajo costo.

El LM35 no requiere de circuitos adicionales para calibrarlo externamente. La baja impedancia de salida, su salida lineal y su precisa calibración hace posible que este integrado sea instalado fácilmente en un circuito de control. Debido a su baja corriente de alimentación se produce un efecto de auto calentamiento muy reducido. Se encuentra en diferentes tipos de encapsulado, el más común es el TO-92, utilizada por transistores de baja potencia.

Además, para poder adaptar la tensión de salida, al valor máximo y mínimo de la entrada analógica de la placa Freescale, se ha realizado un circuito de acondicionamiento de señal (CAS). Se utilizó con un rango entre 0°C hasta 100°C, lo cual provoca una tensión mínima de 0V hasta un valor máximo de 1V. Lo que se busca es que la tensión mínima sea de 0V pero que la máxima sea de 3.33V, que es el nivel máximo de tensión “alto”. Por lo cual se realizó un circuito simple que tiene una ganancia de 3.33 veces, en una configuración no inversora. El amplificador operacional elegido es el LM358 debido a que tiene un bajo consumo.

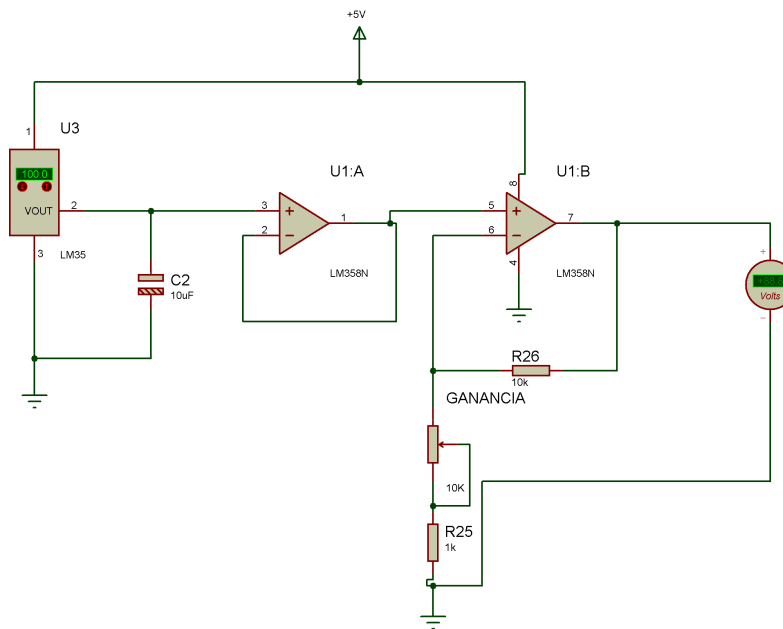


Figura 4.15: CAS.

4.7. Puente H

A continuación se muestra el circuito empleado para controlar el sentido de giro de los motores de CC .Los motores a controlar son de mediana potencia, debido a la limitación de los transistores.

Unidades	Designación	Tipo	Componente
2	2N2222	NPN	Q_1, Q_6
2	TIP32	PNP	Q_2, Q_4
2	TIP31	NPN	Q_3, Q_5
2	1K Ω	1/4 W	$R_1 = R_4$
2	27 Ω	1/4 W	$R_2 = R_3$
4	1N4007	-	$D_1 = D_2 = D_3 = D_4$

Cuadro 4.3: Componentes para un solo motor

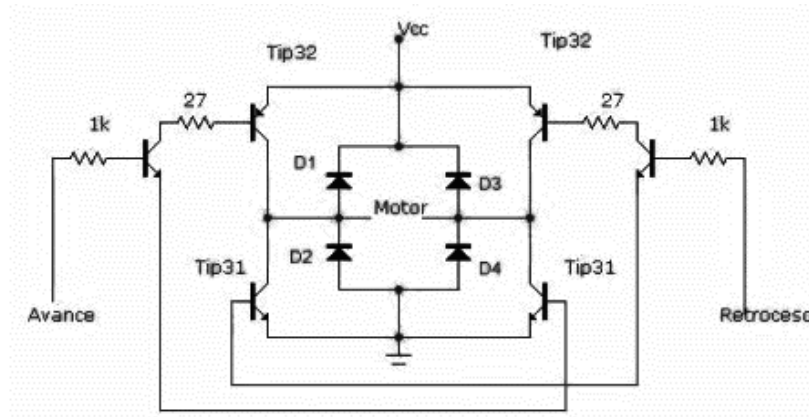


Figura 4.16: Puente H

4.7.1. Funcionamiento del Puente H

El sentido de giro del motor depende de los niveles de tensión que existen en los puntos del circuito etiquetados como “Avance” y “Retroceso”.

Si el nivel de tensión en “Avance” tiene un nivel alto y “retroceso” un nivel bajo, se satura al transistor Q_1 , que a su vez hace entrar en saturación a los transistores Q_2 y Q_5 . Estos dos transistores permiten la circulación de corriente por el motor en un sentido “derecho”.

Si el nivel de tensión en “Retroceso” está en un nivel más alto, se satura el transistor Q_6 , que a su vez hace entrar en saturación a los transistores Q_3 y Q_4 . Estos dos transistores permiten la circulación de corriente por el motor de DC en sentido contrario “Izquierdo”.

Los diodos se implementan como protección de los transistores, debido al cambio de polaridad en las bobinas del motor.

Como se dijo, cada *Puente H* posee 2 entradas y a la salida está conectado el motor, respondiendo a la siguiente tabla de verdad:

Entada 1	Entrada 2	Movimiento del motor
1	1	Frenado
1	0	Avance
0	1	Retroceso
0	0	Frenado

Cuadro 4.4: Lógica del Puente H

4.8. Pantalla LCD 1602A

La siguiente imagen es ilustrativa, pero los pines tienen la misma designación que hemos usado en el proyecto.

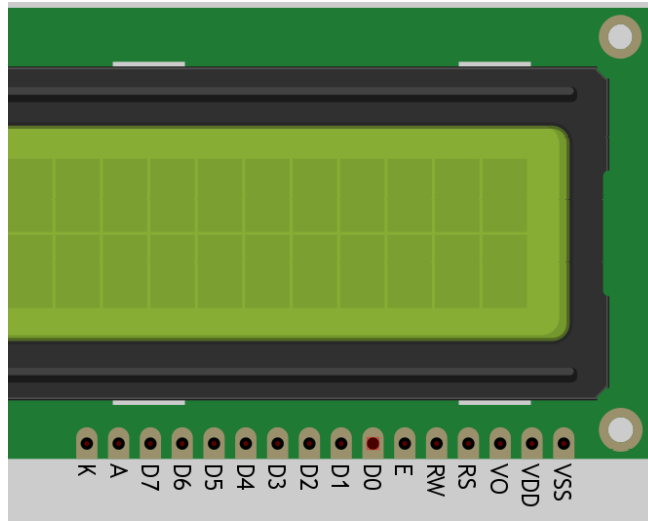


Figura 4.17: LCD Implementado

Pin	Descripción	Observaciones
V_{SS}	Pin negativo ó GND.	Conectado a masa.
V_{DD}	Alimentación principal de la pantalla,	Se alimenta con 5 volt, se recomienda poner en serie una resistencia de 220Ω para evitar daños.
V_O	Es el contraste de la pantalla.	Se opto por conecta una resistencia de 100Ω para fijar dicho contraste. Se puede implementar un potenciómetro preferentemente de $10\ K\Omega$.
RS	Es el selector de registro.	El microcontrolador le comunica al LCD si quiere mostrar caracteres o si lo que quiere es enviar comando de control , como cambiar la posición del cursor por ejemplo.
RW	Comanda la lectura/escritura.	1 para escribir 0 para leer. En este caso siempre estará en cero (conectado a GND).
E	Habilita la pantalla para recibir información.	Lo maneja la placa ARM.
$D_0 - D_7$	Bus de Datos	$D_0 - D_3$ No se han utilizado. La pantalla tiene un bus de dato de 8 bits, pero sólo se usaron 4 bit, del D_4 a D_7 , que servirán para establecer las lineas de comunicaciones.
A, K	Son los pines de luz de fondo de la pantalla (Backlight)	A se conecto a 5 volt mediante un resistor de $1k$ y K a GND.

Cuadro 4.5: Pines del display LCD

4.8.1. Parámetros Máximos Absolutos de Funcionamiento

Parámetros	Simbolo	Min	Max	Unidad
Alimentación	$V_{DD} - V_{SS}$	0	7.0	V
Temperatura de operación	T_{op}	0	+50	°C
Temperatura de Almacenamiento	T_{ST}	-20	+60	°C

Cuadro 4.6: Parámetros Máximos Absolutos

4.8.2. Características Eléctricas

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Supply voltage for LCD	$V_{DD}-V_O$	$T_A=25^{\circ}C$	-	4.6	-	V
Input voltage	V_{DD}		4.7	-	5.5	V
Supply current	I_{DD}	$V_{DD}=5.0V; T_A=25^{\circ}C$	-	1.5	2.5	mA
Input leakage current	I_{LKG}		-	-	1.0	μA
"H" level input voltage	V_{IH}		2.2	-	V_{DD}	V
"L" level input voltage	V_{IL}	Twice initial value or less	0	-	0.6	V
"H" level output voltage	V_{OH}	$I_{OH}=-0.25mA$	2.4	-	-	V
"L" level output voltage	V_{OL}	$I_{OL}=1.6mA$	-	-	0.4	V
Backlight supply power	V_F		-	4.2	4.5	V

Figura 4.18: Características Electricas

4.8.3. Código

La siguiente descripción es la necesaria para poder implementar el LCD utilizando la librería "TextLCD.h" que proporciona el compilador en línea "mbed" utilizado.

TextLCD Class Reference

```
#include <TextLCD.h>
```

Public Types

```
enum LCDType { LCD16x2, LCD16x2B, LCD20x2, LCD20x4 }
```

LCD panel format.

[More...](#)

Public Member Functions

[TextLCD](#) (PinName rs, PinName e, PinName d4, PinName d5, PinName d6, PinName d7, [LCDType](#) type=LCD16x2)

Create a **TextLCD** interface.

int [putc](#) (int c)

Write a character to the LCD.

int [printf](#) (const char *format,...)

Write a formatted string to the LCD.

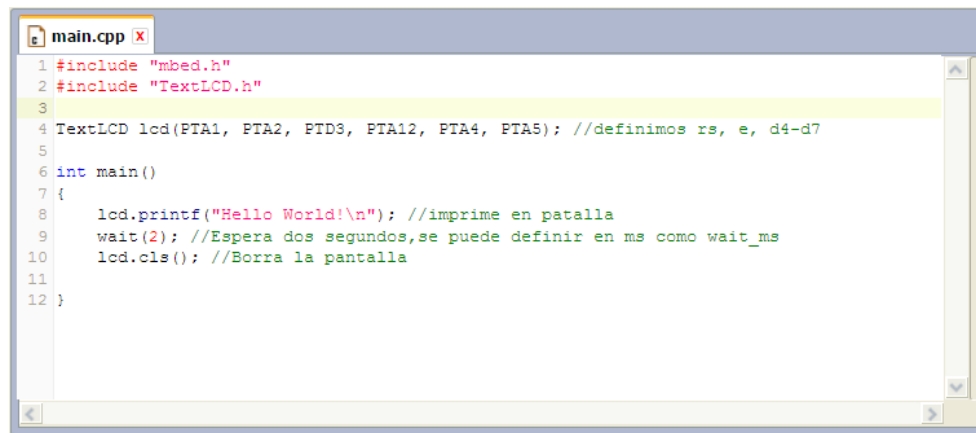
void [locate](#) (int column, int row)

Locate to a screen column and row.

void [cls](#) ()

Clear the screen and locate to 0,0.

Figura 4.19: Clase TextLCD



```
main.cpp x
1 #include "mbed.h"
2 #include "TextLCD.h"
3
4 TextLCD lcd(PTA1, PTA2, PTD3, PTA12, PTA4, PTA5); //definimos rs, e, d4-d7
5
6 int main()
7 {
8     lcd.printf("Hello World!\n"); //imprime en pantalla
9     wait(2); //Espera dos segundos, se puede definir en ms como wait_ms
10    lcd.cls(); //Borra la pantalla
11
12 }
```

Figura 4.20: Fragmento ilustrativo del código

Circuito General

Este proyecto posee una placa central en la cual se encuentra el procesador a utilizar y las conexiones de los distintos sensores y componentes del proyecto como display y motores.

Esta placa central está alimentada por una batería de 12V, tensión que internamente es modificada para los diversos usos. A continuación se muestran los esquemáticos:

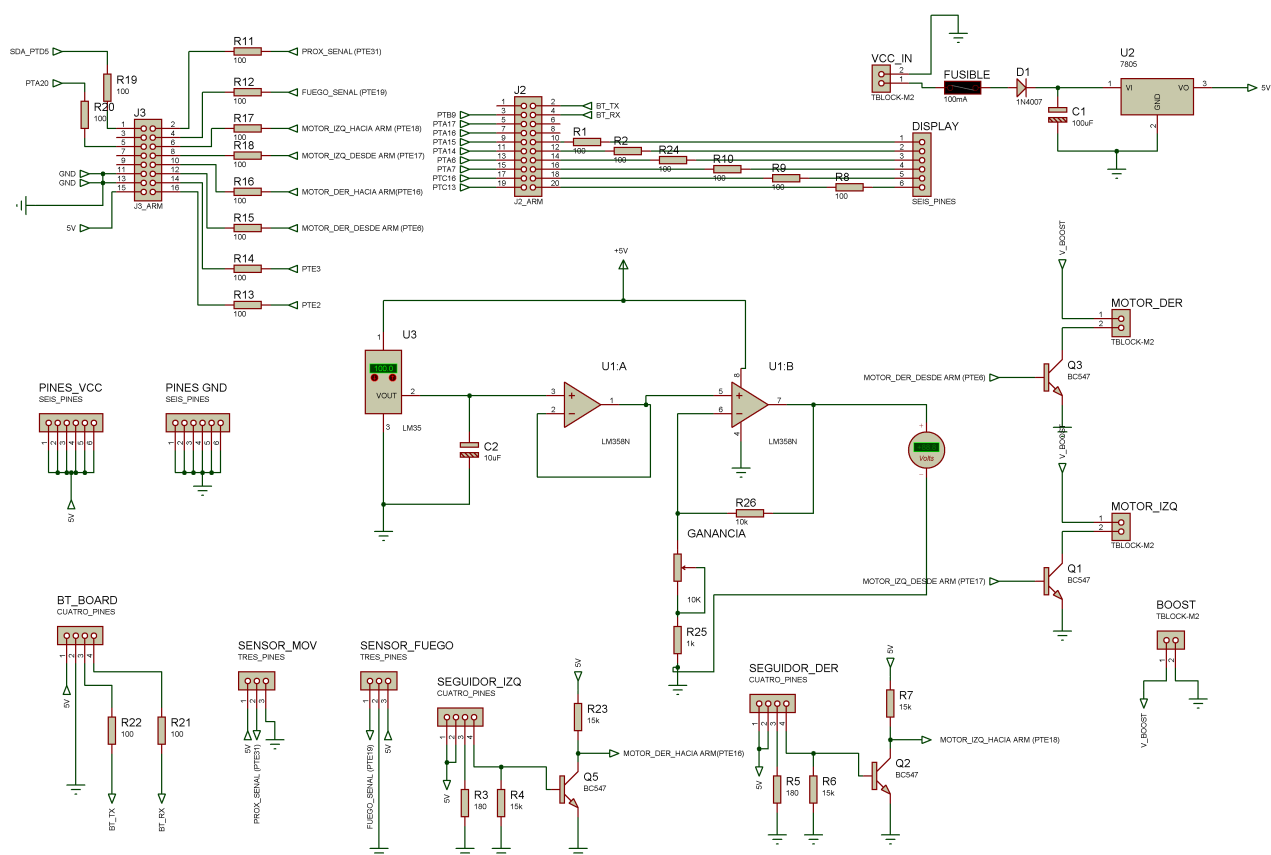


Figura 5.1: Diagrama de placa

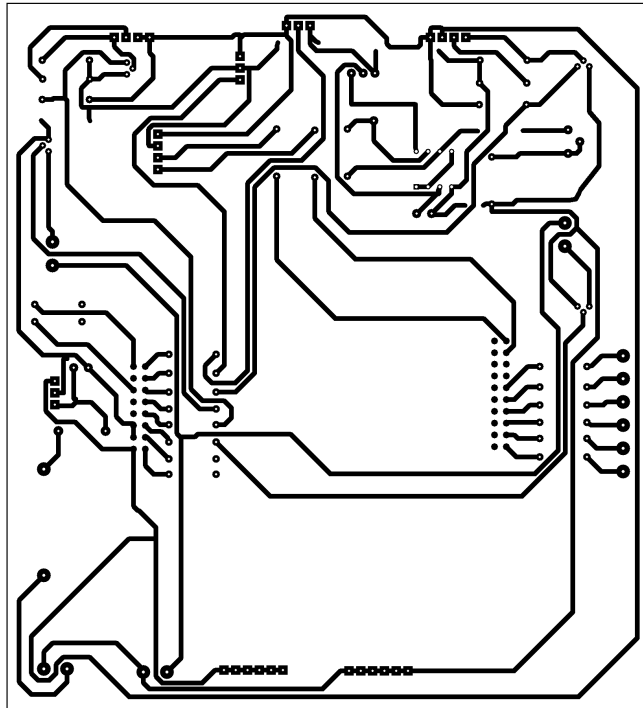


Figura 5.2: PCB general

Capítulo 6

Programación

6.1. Código

El código introducido al procesador ARM es el encargado de ensamblar todas las partes de este proyecto. Mediante este se diseña el funcionamiento y las decisiones a tomar según las circunstancias. A continuación se presenta el código utilizado, luego se analizarán los bloques más importantes. Se utilizó la plataforma MBED ([link](#)) que permite programar de forma online, prácticamente sin tener que instalar nada, solamente un firmware para la placa y un driver para la computadora para poder transferir nuestro código a la memoria de la placa. En este caso se utilizó el lenguaje C y algunas librerías que nos brinda esta plataforma.

```
#include "RawSerial.h"
#include "mbed.h"
#include "string.h"
#include "TextLCD.h"
#define MAX_VUELTAS 2

void display ();
void enviar_bt ();
void sensor ();
void modo_automatico ();
float temperatura();

DigitalOut led_verde(LED_GREEN);
DigitalOut led_rojo(LED_RED);
DigitalIn Sensor_Fuego(PTE19);
DigitalIn Sensor_Mov(PTE31);
DigitalIn Motor_Izq_IN(PTE18);
DigitalOut Motor_Izq_adelante_OUT(PTE17);
DigitalOut Motor_Izq_retroceso_OUT(PTE3);
DigitalIn Motor_Der_IN(PTE16);
DigitalOut Motor_Der_adelante_OUT(PTE6);
DigitalOut Motor_Der_retroceso_OUT(PTE2);
AnalogIn entrada_temp(PTB0);
RawSerial pc(PTE0, PTE1); //TX,RX
TextLCD lcd(PTA13, PTD2, PTD4, PTD6, PTD7, PTD5); //RS,E,D4-D7
int modo_v = 1, vueltas = 0, eventos = 0, valor = 0;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

[illegible]

```

break;}

////////////////////////////////////

case 50:{ //Ingresa la opcion dos
lcd.cls();
lcd.printf("Modo MANUAL.\n");
pc.putc(13);
pc.printf("-> MODO MANUAL \n");
do{
mov=pc.getc();

switch(mov){
case 97:{ //a //los caracteres estan escritos en codigo ASCII
while(mov==97){
lcd.cls();
lcd.printf("GIRO IZQUIERDA."); //GIRO IZQUIERDA : Motor derecho ↑ Motor izquierdo ↓
Motor_Der_adelante_OUT = 1;
Motor_Der_retroceso_OUT = 0;
Motor_Izq_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 1;
mov=pc.getc();}
break;}

case 119:{ //w
while(mov==119){
lcd.cls();
lcd.printf("AVANCE."); //ADELANTE : Motor derecho ↑ Motor izquierdo ↑
Motor_Der_adelante_OUT = 1;
Motor_Der_retroceso_OUT = 0;
Motor_Izq_adelante_OUT = 1;
Motor_Izq_retroceso_OUT = 0;
mov=pc.getc();}
break;}

case 100:{//d //GIRO DERECHA: Motor derecho ↓ Motor izquierdo ↑
while(mov==100){
lcd.cls();
lcd.printf("GIRO DERECHA.");
Motor_Der_adelante_OUT = 0;
Motor_Der_retroceso_OUT = 1;
Motor_Izq_adelante_OUT = 1;
Motor_Izq_retroceso_OUT = 0;
mov=pc.getc();}
break;}

case 115:{//s //ATRAS : Motor derecho ↓ Motor izquierdo ↓
while(mov==115){
lcd.cls();
lcd.printf("RETROCESO.");
Motor_Der_adelante_OUT = 0;
Motor_Der_retroceso_OUT = 1;

```

```
Motor_Izq_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 1;
mov=pc.getc();}
break;}
```

```
case 127:{//q sig=127; //MOTOR DETENIDO
lcd.cls();
lcd.printf("DETENIDO.");
Motor_Der_adelante_OUT = 0;
Motor_Der_retroceso_OUT = 0;
Motor_Izq_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 0;
break;}
```

```
default: { //MOTOR DETENIDO
lcd.cls();
lcd.printf("DETENIDO.");
Motor_Der_adelante_OUT = 0;
Motor_Der_retroceso_OUT = 0;
Motor_Izq_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 0; //apague los dos motores
break;}
}
}while(sig!=127);
break; }
```

```
case 51:{ //opcion tres
pc.putc(13);
pc.printf("\n\n A elegido Datos\n");
senzar();
enviar_bt ();
pc.putc(13);
pc.printf(" Pulse cualquier tecla para continuar...\n");
pc.getc();
lcd.cls();
break;}
```

[illegible]

```

pc.printf(" 3_ OBTENER DATOS \n");
pc.putc(13);
elec=pc.getc();
pc.putc(elec);
enter=pc.getc(); }

while(52<elec||enter!=127||elec<49); break; }
} //FIN switch
eventos = 0;
} //FIN while
}

void display (){
lcd.cls()
if (valor == 0)
lcd.printf("Situacion controlada\n");
if (valor == 1)
lcd.printf("Movimiento detectado\n");
if (valor == 2)
lcd.printf("Fuego detectado\n");
if (valor == 3)
lcd.printf("Fuego y Mov. detectado\n");
}

void enviar_bt (){
int x = 0;
float valor_temperatura;
pc.putc(13);
pc.printf("\n\n");
pc.putc(13);
pc.printf(" Transmitiendo estado... \n");
pc.putc(13);
pc.printf(" Se registraron %d eventos en la vuelta %d:\n ", eventos, vueltas);
pc.putc(13);

if (valor == 0)
pc.printf(" Situacion controlada. \n");
if (valor == 1)
pc.printf(" Movimiento detectado. \n");
if (valor == 2)
pc.printf(" Fuego detectado. \n");
if (valor == 3)
pc.printf(" Fuego y movimiento detectado. \n");
pc.putc(13);
valor_temperatura = temperatura();
pc.printf(" La temperatura es de %.3f grados. \n\n", valor_temperatura);

for (x = 0; x < vueltas; x++)
led_verde = !led_verde;
wait_ms(1000);
led_verde = 0;
}

void sensar (){
static int ok = 0, movimiento = 1, fuego = 2, alarma = 3;
lcd.cls();

```

```

led.printf("Sensando...");
if ((Sensor_Fuego == 1) && (Sensor_Mov == 0)) // OK
{valor = ok; wait_ms(2500);}
if ((Sensor_Fuego == 0) && (Sensor_Mov == 0)) // SI HAY FUEGO
{valor = fuego; wait_ms(2500); eventos += 1;}
if ((Sensor_Fuego == 1) && (Sensor_Mov == 1)) // SI HAY MOVIMIENTO
{valor = movimiento; wait_ms(2500); eventos += 1;}
if ((Sensor_Fuego == 0) && (Sensor_Mov == 1)) // SI HAY FUEGO Y MOVIMIENTO
{ valor = alarma; wait_ms(2500); eventos += 1; }
display(); }

```

```

void modo_automático (){

```

```

for ( ; vueltas < MAX_VUELTAS; vueltas++) {
for (;;) {

```

```

if (Motor_Izq_IN == 1 && Motor_Der_IN == 1){
Motor_Izq_adelante_OUT = 1;
Motor_Izq_retroceso_OUT = 0;
Motor_Der_adelante_OUT = 1;
Motor_Der_retroceso_OUT = 0;}

```

```

if (Motor_Izq_IN == 1 && Motor_Der_IN == 0){
Motor_Izq_adelante_OUT = 1;
Motor_Izq_retroceso_OUT = 0;
Motor_Der_adelante_OUT = 0;
Motor_Der_retroceso_OUT = 1;}

```

```

if (Motor_Izq_IN == 0 && Motor_Der_IN == 1){
Motor_Izq_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 1;
Motor_Der_adelante_OUT = 1;
Motor_Der_retroceso_OUT = 0;}

```

```

if (Motor_Der_IN == 0 && Motor_Izq_IN == 0) {
// si se detecta una franja horizontal, espera y envia el estado por bluetooth
Motor_Izq_adelante_OUT = 0;
Motor_Der_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 0;
Motor_Der_retroceso_OUT = 0;
wait_ms(250);
pc.putc(13);
sensar();
enviar_bt ();
pc.putc(13);
pc.printf(" Vehiculo en movimiento, Vuelta %d \n\n", vueltas);

```

```

for (int x=0; x < 30; x++){
led_rojo = !led_rojo;
wait_ms(50);}
Motor_Izq_adelante_OUT = 1;
Motor_Der_adelante_OUT = 1;
Motor_Izq_retroceso_OUT = 0;
Motor_Der_retroceso_OUT = 0;
wait_ms(500);
break; // Si detecta una linea, interrumpe el conteo de treinta segundos } //cierro if

```

```

} //cierro for (;){

Motor_Izq_adelante_OUT = 0;
Motor_Der_adelante_OUT = 0;
Motor_Izq_retroceso_OUT = 0;
Motor_Der_retroceso_OUT = 0;
vueltas = 0;
lcd.cls();
lcd.printf("Fin modo AUTO\n");
lcd.printf(" %d eventos\n", eventos);
pc.putc(13);
pc.printf(" Pulse cualquier tecla para continuar...\n");
pc.getc();
lcd.cls(); }

```

```

float temperatura (void){
float temp, temp1, temp2, temp3;
temp1 = entrada_temp.read() * 100;
wait_ms(100);
temp2 = entrada_temp.read() * 100;
wait_ms(100);
temp3 = entrada_temp.read() * 100;
wait_ms(100);
temp = (temp1+temp2+temp3) / 3;
return temp; }

```

6.2. Análisis

Se comenzará con las **librerías** incluidas, estas son : Rawserial, para la comunicación serial que se realiza con el módulo Bluetooth; luego la librería propia del compilador mbed.h; string.h se ocupa de brindar funciones para manejar cadenas de caracteres; TextLCD.h permite el manejo del display externo que tiene el auto. La constante MAX_VUELTAS, permite setear el número de vueltas que hace el auto en el modo automático antes de regresar al menú principal.

A continuación se definen las **funciones** utilizadas en el programa: la primera es para mostrar texto en el display, cuando se detecta algun movimiento y/o fuego, otra función para enviar por bluetooth los datos relevados, la función sensar releva los datos de los sensores, el modo_ automatico permite que el auto recorra la trayectoria establecida en la pista con el fondo blanco y la franja negra en el centro, aquí el auto hace uso de los sensores CNY70. Por ultimo, la función temperatura, realiza tres lecturas de temperatura y hace el promedio de las mismas.

Los **pines** de entrada y de salida de la placa pueden ser de tipo digital o de tipo analógico. Sólo para la salida del sensor LM35 se utiliza una entrada analógica, el resto son digitales. También se definen los pines correspondientes al display.

La función main, es relativamente simple, muestra el mensaje inicial en el display del auto y luego, brinda el menu con las opciones que aparecerán en la aplicación del celular. La opción 1 llama a la función modo_ automatico. En la opción 2, se dan las instrucciones para el modo manual, dando las instrucciones necesarias para el manejo de los motores a través del puente H. La opción 3, realiza un censado "manual" en el momento que se desee.

Capítulo 7

Conclusión

Es notorio que “ARMando” es un prototipo de auto de seguridad, pero su función fue poder iniciar en algunos temas de robótica a los alumnos que realizamos este trabajo.

Se descubrió que existen puntos claves a evaluar cuando se decide realizar proyectos en esta rama de la electrónica:

- **Objetivo:** tener bien en claro a que se quiere llegar es fundamental, ya que cuando se comienza a trabajar es fácil "irse por las ramas" como se dice comúnmente. En un punto esto es bueno porque abre nuevas puertas, pero debe controlarse para llegar al fin deseado.
- **Centro de comando:** es necesario poner un punto central al cual unir todas las partes del proyecto. Se puede trabajar en bloques separados, pero luego hay que agregar cada parte a un enlazador que unifique el proyecto y lo haga una sola cosa. En este proyecto esta función la cumplió el procesador ARM y el código embebido en él.
- **Programación:** para poder tener un centro de comando y de ejecución, es necesario saber dar las órdenes. Este es el trabajo del código desarrollado, recibe información, analiza, y produce un acción.
- **Cálculos:** es casi imposible armar un circuito eficiente y preciso sin hacer cálculos matemáticos. Primero se debe adquirir conocimiento y luego hacer.
- **Movilidad:** esto incluye rozamiento, engranajes, potencia y giros. En un principio parece sencillo, pero cuando se comienza a trabajar aparecen variables que no se consideran en un principio.
- **Conexión inalámbrica:** este tema posee muchas variantes y posibilidades. Por esta razón es necesario elegir un tipo de conexión según la función a desarrollar e investigar cómo llevar a cabo esa forma. En este caso se optó por utilizar Bluetooth que es uno de los más sencillos.

Existen muchas variables a tener en cuenta, pero también se pueden ampliar si se desea. Como ya se dijo, el vehículo de seguridad “ARMando” es un prototipo, y por lo tanto está disponible a cambios y mejoras múltiples, según el fin para el que sea usado.