



**ESCUELA SUPERIOR POLITÉCNICA DEL  
LITORAL  
FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y  
COMPUTACIÓN**

“Control de Robot mediante mensajes de texto utilizando el  
dispositivo Narobo DroneCell en interfaz con el Pololu 3π.”

**TESINA DE SEMINARIO**

**Previa a la obtención del Título de:**

**INGENIERO EN ELECTRÓNICA Y  
TELECOMUNICACIONES.**

**Presentado por**

**JAIME ISRAEL IZQUIERDO VALLADAREZ  
OTHON ANDRES PONCE ALVARADO**

Guayaquil – Ecuador

2011

# AGRADECIMIENTO

A Dios por permitirme alcanzar esta meta y darme una familia con la que puedo contar siempre.

Al Ing. Carlos Valdivieso, por su guía y respaldo en la elaboración del presente trabajo.

El agradecimiento especial es para las personas que siempre han estado a mi lado sin importar la distancia, mi familia, que es el pilar vital de mi existencia.

*Jaime Israel Izquierdo Valladarez*

Agradezco a Dios ya que es el único que nos puede dar la vida, salud, fuerza y sabiduría para poder salir adelante superando cualquier reto que se nos presente durante nuestra carrera como profesionales.

A mi madre y a mis hermanos que fueron los que siempre estuvieron junto a mí y me apoyaron en cualquier problema que se me haya presentado en mi vida.

*Othón Andrés Ponce Alvarado*

# DEDICATORIA

A Dios que sin su bendición nada se hubiese podido realizar.

A mis padres Luis Izquierdo Pacheco y Rosa Valladarez Guamán, que con su apoyo incondicional y la muestra del gran amor que tengo hacia ellos se realizó éste importante logro en mi vida. También a mis hermanas Shirley, Joana, Fernanda y mi hermano Luis, que siempre me han brindado su apoyo.

*Jaime Israel Izquierdo Valladarez*

Este trabajo se lo dedico a Dios, nuestro Padre Todo Poderoso ya que sin su ayuda y sin su bendición, nada podríamos haber logrado durante toda nuestra carrera.

También le dedico este trabajo a mi querida madre SARA ALVARADO ya que sin su ayuda y sin su incondicional apoyo nunca hubiese podido llegar a tan importante momento de mi vida como es este.

*Othón Andrés Ponce Alvarado*

# TRIBUNAL DE SUSTENTACIÓN

---

Ing. Carlos Valdivieso A.

Profesor del Seminario de Graduación

---

Ing. Hugo Villavicencio V.

Profesor delegado del Decano

# DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta Tesina, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”

(Reglamento de Graduación de la ESPOL).

---

Jaime Israel Izquierdo Valladarez

---

Othón Andrés Ponce Alvarado

# RESUMEN

El presente trabajo consiste diseñar un código para controlar un Robot mediante mensajes de texto utilizando el dispositivo Narobo DroneCell en interfaz con el Pololu 3π.

El código se lo realizará utilizando el programa AVR Studio el cual permite diseñar cualquier tipo de código ya sea en lenguaje C o ASM y la compilación se la utiliza para microcontroladores ATMEL.

El funcionamiento del código se lo comprobará mediante la herramienta de simulación llamada PROTEUS ya que es un programa de mucha utilidad para simular cualquier microcontrolador.

Nuestro proyecto se basa básicamente en la investigación del dispositivo Narobo DroneCell y el Pololu 3π. El Narobo DroneCell se puede resumir en un dispositivo inalámbrico capaz de soportar llamadas, mensajes de texto y acceso a internet, el Pololu 3π es un Robot programable que permite realizar muchas funciones para diferentes proyectos especialmente como seguidor de línea y nuestro objetivo es controlar los movimientos del Pololu 3π por medio de mensajes de texto para lo cual usaremos las características del Narobo DroneCell de soportar mensajes de texto, y éste texto será transmitido al Pololu 3π a través del puerto serial.

En operación normal usaremos un celular que enviará instrucciones por medio de mensajes de texto al Narobo DroneCell, el cual recibe y trasmite el mensaje de texto en forma serial al Pololu 3π.



# ÍNDICE GENERAL

AGRADECIMIENTO .....	II
DEDICATORIA .....	IV
TRIBUNAL DE SUSTENTACIÓN.....	VI
DECLARACIÓN EXPRESA .....	VII
RESUMEN .....	VIII
ÍNDICE GENERAL .....	IX
ÍNDICE DE FIGURAS .....	XIII
INTRODUCCIÓN .....	XV
CAPÍTULO 1 .....	1
DESCRIPCIÓN GENERAL DEL PROYECTO .....	1
1.1 ANTECEDENTES .....	1
1.2 DESCRIPCIÓN DEL PROYECTO.....	2
1.3 APLICACIÓN .....	3
1.4 DESCRIPCIÓN DEL PROBLEMA.....	4
1.5 PROYECTOS SIMILARES.....	5
1.5.1 PIC Based, Obstacle-Avoiding Robot.....	5
1.5.2 PIC- Based, Obstacle - Avoiding Robot.....	6

1.5.3	AIRbot.....	7
CAPÍTULO 2 .....		8
FUNDAMENTO TEÓRICO .....		8
2.1	HERRAMIENTAS DE SOFTWARE .....	8
2.1.1	AVR STUDIO 4.....	8
2.1.1.1	Ventana Principal .....	10
2.1.1.2	Ventana de Registros.....	11
2.1.1.3	Ventana de Memoria .....	11
2.1.1.4	Ventana de Mensajes. ....	12
2.1.1.5	Características del AVR STUDIO.....	12
2.1.2	PROTEUS 7.7.....	13
2.1.2.1	ISIS.- Intelligent Schematic Input System .....	14
2.1.2.2	ARES. - Advanced Routing and Editing Software.....	14
2.2	Herramientas de Hardware .....	15
2.2.1	POLOLU 3π .....	15
2.2.1.1	ATmega328P .....	16
2.2.2	NAROBO DRONECELL.....	19
2.2.2.1	Características de la tarjeta Narobo DroneCell .....	20
2.3	Transmisión de Mensajes vía celular.....	22

2.3.1	Teléfonos GSM .....	22
2.3.2	Lenguaje de Comunicación .....	22
2.3.2.1	Comandos .....	22
CAPÍTULO 3 .....		24
DESCRIPCIÓN E IMPLEMENTACIÓN DEL PROYECTO .....		24
3.1	Diseño Preliminar .....	24
3.1.1	Comunicación entre Celular y Narobo DroneCell .....	25
3.1.2	Transmisión de Datos Serial .....	26
3.1.3	Ejecución de los movimientos y formato del mensaje SMS .....	27
3.1.4	Implementación Física .....	28
3.2	Descripción del proyecto final .....	29
3.2.1	Bienvenida .....	29
3.2.2	Establecer la comunicación .....	29
3.3	Diagramas de Bloques del Proyecto .....	30
3.3.1	Diagrama de bloques de la primera etapa .....	30
3.3.2	Diagrama de bloques de la segunda etapa .....	30
3.3.3	Diagrama de bloques de la tercera etapa .....	31
3.3	Diagrama de flujo principal .....	32
CAPÍTULO 4 .....		33

SIMULACION Y PRUEBAS.....	33
4.1 Pruebas en Hyper Terminal.....	33
4.1.1 Prueba de Transmisión.....	33
4.1.2 Prueba de Recepción .....	34
4.2 Simulación en Proteus 7.7.....	35
4.3 Resultados Experimentales .....	40
CONCLUSIONES .....	1
RECOMENDACIONES.....	3
ANEXOS .....	5
BIBLIOGRAFIA.....	30

# ÍNDICE DE FIGURAS

<i>FIGURA 1-1: Funcionamiento Básico del Proyecto</i> .....	2
<i>FIGURA 1-2: RC 3π</i> .....	5
<i>FIGURA 1-3: PIC-Based, Obstacle – Avoiding Robot</i> .....	6
<i>FIGURA 1-4: AIRbot</i> .....	7
<i>FIGURA 2-1: ENTORNO GRÁFICO DEL AVR STUDIO</i> .....	8
<i>FIGURA 2-2: VENTANA PRINCIPAL DEL AVR STUDIO</i> .....	10
<i>FIGURA 2-3: VENTANA DE REGISTROS</i> .....	11
<i>FIGURA 2-4: VENTANA DE MEMORIA</i> .....	11
<i>FIGURA 2-5: VENTANA DE MENSAJES</i> .....	12
<i>FIGURA 2-6: ENTORNO GRÁFICO DE PROTEUS V.7.7</i> .....	13
<i>FIGURA 2-7: POLOLU 3π</i> .....	15
<i>FIGURA 2-8: ATmega328P</i> .....	18
<i>FIGURA 2-9: Tabla comparadora del ATmega328P con otros ATmegas</i> .....	18
<i>FIGURA 2-10: Narobo DroneCell</i> .....	20
<i>FIGURA 2-11: Pines del Narobo DroneCell</i> .....	21
<i>FIGURA 3-1: Componentes de la primera etapa del proyecto</i> .....	25
<i>FIGURA 3-2: Componentes de la segunda etapa</i> .....	26
<i>FIGURA 3-3: Componentes de las etapas del proyecto</i> .....	28
<i>FIGURA 3-4: Diagrama de bloques primera etapa</i> .....	30
<i>FIGURA 3-5: Diagrama de bloques segunda etapa</i> .....	30
<i>FIGURA 3-6: Diagrama de bloques tercera etapa</i> .....	31
<i>FIGURA 3-7: Diagrama de flujo del proyecto</i> .....	32

<i>FIGURA 4-1 Prueba de Transmisión.....</i>	<i>33</i>
<i>FIGURA 4-2 Prueba de Recepción.....</i>	<i>34</i>
<i>FIGURA 4-3 Iniciación del Proyecto Simulado .....</i>	<i>35</i>
<i>FIGURA 4-4 Press B to Start en Proteus.....</i>	<i>36</i>
<i>FIGURA 4-5 Iniciando DroneCell .....</i>	<i>37</i>
<i>FIGURA 4-6 DroneCell Listo .....</i>	<i>38</i>
<i>FIGURA 4-7 Error text.....</i>	<i>39</i>
<i>FIGURA 4-8 Press B to Start “experimental” .....</i>	<i>40</i>
<i>FIGURA 4-9 Espero el SMS.....</i>	<i>41</i>
<i>FIGURA 4-10 Espero Nuevo SMS .....</i>	<i>41</i>

# INTRODUCCIÓN

En la actualidad existen Robots controlados por el hombre y son de mucha utilidad para investigaciones como por ejemplo Robots que se utilizan para explorar otros planetas, los grandes países desarrollados en tecnología tienen muchas expectativas con el uso de dispositivos para controlar Robots, especialmente con la conexión inalámbrica y el uso de los microcontroladores.

En este presente trabajo se presentará la arquitectura, funcionamiento y programación del Pololu 3 $\pi$  además de la información adicional investigada por medio de la herramienta de internet como los diferentes proyectos básicos realizados en el desarrollo de la materia.

El Narobo DroneCell es otro dispositivo de investigación para el proyecto y de mucha utilidad ya que sin este dispositivo no tendríamos la conexión adecuada para realizar las instrucciones que le enviemos a nuestro Robot como son la de dirigirse hacia adelante, hacia atrás, dirigirse hacia la izquierda o derecha y otras instrucciones adicionales que nos propongamos realizar y sacar provecho del DroneCell, sus características se las mostrará en el desarrollo de los capítulos posteriores.

Finalmente, después de haber diseñado nuestro código se realizará las pruebas específicas para comprobar el funcionamiento de nuestro diseño, haciendo el uso de PROTEUS y luego verificándolo en nuestro Robot.

# CAPÍTULO 1

## DESCRIPCIÓN GENERAL DEL PROYECTO

### 1.1 ANTECEDENTES

Los microcontroladores tienen sus raíces en el desarrollo de la tecnología de los circuitos integrados, los cuales también son conocidos como micro-computadoras ya que poseen las funcionalidades de entrada, procesamiento y salida de datos. Con un microcontrolador se pueden realizar las mismas funciones que se obtendrían con varios elementos TTL como por ejemplo sumar, comparar, almacenamiento de datos, etc.

Nuestro trabajo se basa en el estudio de los microcontroladores como son el AT90S1200, ATtiny2313 y el ATmega169 los cuales nos han servido de mucho para tener conocimientos y empezar a trabajar con el Pololu 3 $\pi$  ya que el corazón del Pololu es el ATmega328P y éste microcontrolador tiene características similares a los microcontroladores ya mencionados.



## 1.2 DESCRIPCIÓN DEL PROYECTO

El presente trabajo se enfoca en el control de los movimientos del Robot Pololu 3π mediante mensajes de texto enviados desde un dispositivo móvil (Celular), el cual va a ser recibido por el Narobo DroneCell, ésta tarjeta va a tener incorporado un chip GSM estableciendo la comunicación entre el celular y el Pololu 3π.

Con la ayuda del Narobo DroneCell se establecerá una comunicación serial con el Robot Pololu 3π para poder transmitirle el mensaje que fue enviado desde el celular.

Una vez que el mensaje ha sido transmitido al Pololu 3π, éste se encargará de reconocer el contenido y comparar con cadenas predefinidas dentro del programa para que de esta manera pueda obtener la instrucción adecuada y ejecutar el respectivo movimiento.

A continuación se presenta un diagrama donde se muestra el funcionamiento básico de nuestro proyecto.

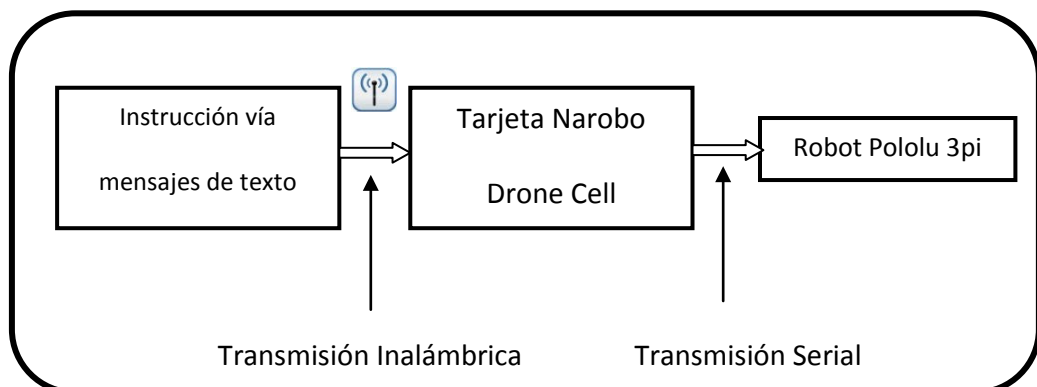


FIGURA 1-1: Funcionamiento Básico del Proyecto

### **1.3 APLICACIÓN**

Los microcontroladores tienen muchas aplicaciones ya que hoy en día la tecnología se expande sin límites, estos se encuentran en todos los circuitos de grandes aplicaciones como computadoras, Reuters, Robots, etc.

Nuestro Pololu 3π tiene incorporado el microcontrolador ATmega328P el cual tiene varias opciones para realizar cualquier proyecto que tenga aplicaciones grandes, posee comunicación serial para interactuar con otros micros y realizar diferentes objetivos.

El Narobo DroneCell como se lo ha detallado tiene la capacidad de recibir o enviar mensajes de texto, llamadas y acceso a internet.

Las aplicaciones principalmente son para interconexiones inalámbricas como por ejemplo:

- Activación o Desactivación de Alarmas ya sea para vehículos o garajes de casa mediante el simple envío de mensajes de texto.
- Manipulación mediante mensajes de texto a maquinarias pesadas en lugares peligrosos para el ser humano, ejemplo: lugares con ambientes tóxicos.
- Competencias de Robots sumos manipulados por mensajes de texto desde cualquier celular móvil.

## **1.4 DESCRIPCIÓN DEL PROBLEMA**

El uso de la comunicación inalámbrica es amplio y tiene muchas aplicaciones, especialmente para la solución de problemas industriales, científicos, etc.

Nuestro proyecto trata de solucionar los problemas que se presentan al momento de manipular maquinarias en lugares peligrosos para el ser humano, como por ejemplo: sitios donde se presenten gases tóxicos.

Otro problema que solucionaría es al momento de controlar Robots que se dediquen a funciones de repartición o traslado de materiales en industrias.

Se puede programar un microcontrolador para que trabaje con el Narobo y controlar sistemas de seguridad como por ejemplo cámaras de seguridad en lugares importantes como bancos, industrias, edificios, etc.

El presente sistema desarrollado puede resolver los problemas ya mencionados y ser usado como guía para trabajar con proyectos más complejos que se puedan presentar al momento de crear soluciones a diferentes sistemas de aplicaciones inalámbricas.

## 1.5 PROYECTOS SIMILARES

### 1.5.1 PIC Based, Obstacle-Avoiding Robot

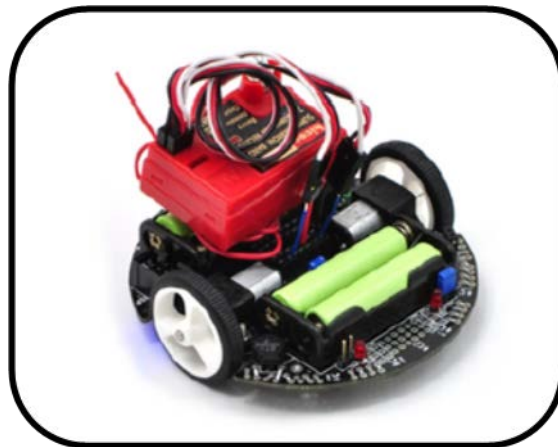


FIGURA 1-2: RC 3π

Por medio de un receptor de Radio Control se puede convertir al Pololu 3π en un Robot teledirigido.

Esta combinación realiza que el Pololu 3π se lo pueda conducir a una velocidad superior a 1m/s. Se debe utilizar un transmisor de radio frecuencia para conducir al Pololu 3π.

Mediante un algoritmo simple y la incorporación de un receptor de RC, el Robot 3π es un Robot controlado por radio frecuencia a una velocidad agradable y una excelente capacidad de giro.

### 1.5.2 PIC- Based, Obstacle - Avoiding Robot

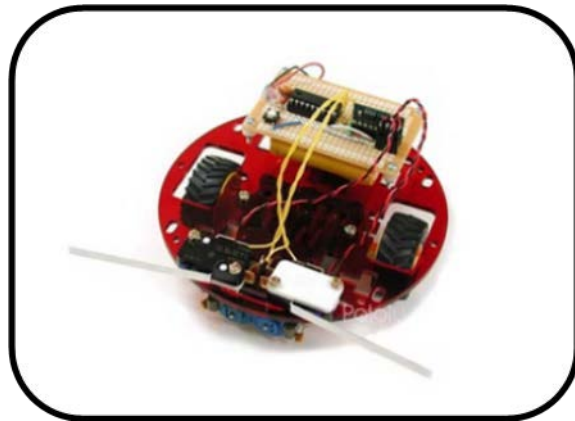


FIGURA 1-3: PIC-Based, Obstacle – Avoiding Robot

Este Robot se basa en el Pololu micro dual serial motor controller con un microcontrolador PIC16F628 de Microchip y su principal objetivo es evitar los obstáculos que se le presentan al momento de realizar sus movimientos.

Este pequeño modulo se lo usa en este proyecto para controlar dos motores, pero se puede conectar en cadena varias unidades para controlar hasta 62 motores con un cable en serie. La alimentación del motor puede ser tan baja como 2V.

La razón principal para usar el PIC16F628 es que tiene incorporado el UART para establecer la comunicación serial con el módulo Pololu micro y controlar los movimientos de los motores.

### 1.5.3 AIRbot

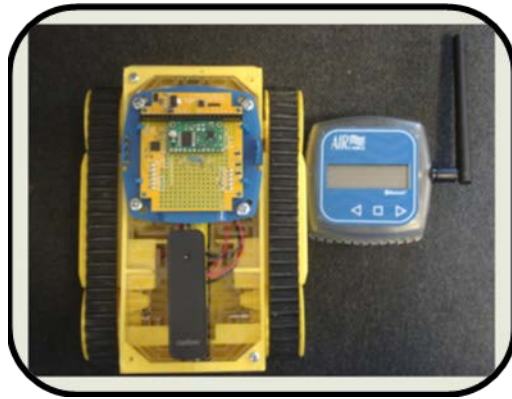


FIGURA 1-4: AIRbot

AIRbot es un Robot que puede ser controlado por un teléfono celular y puede enviar imágenes a teléfonos móviles.

Este demo muestra capacidades para manejar las interrupciones de las líneas digitales, la modulación de la energía, la toma de decisiones y la interconexión móvil.

Utiliza un chasis de orugas RP5 amarilla que es la base del Robot completo con un soporte de batería, dos motores de corriente continua y dos trenes de transmisión independientes, este chasis tiene un controlador de Robot y unos sensores de distancia; también consta de una placa transparente que sirve para adicionar elementos; y un qik 2s9v1 dual serial motor controller que sirve para un fácil y variable manejo de los motores.

# CAPÍTULO 2

## FUNDAMENTO TEÓRICO

En el presente capítulo detallaremos las herramientas utilizadas para el desarrollo de nuestro proyecto como software y hardware, así como también las características y funcionamiento de cada una de las partes involucradas en el proyecto.

### 2.1 HERRAMIENTAS DE SOFTWARE

#### 2.1.1 AVR STUDIO 4

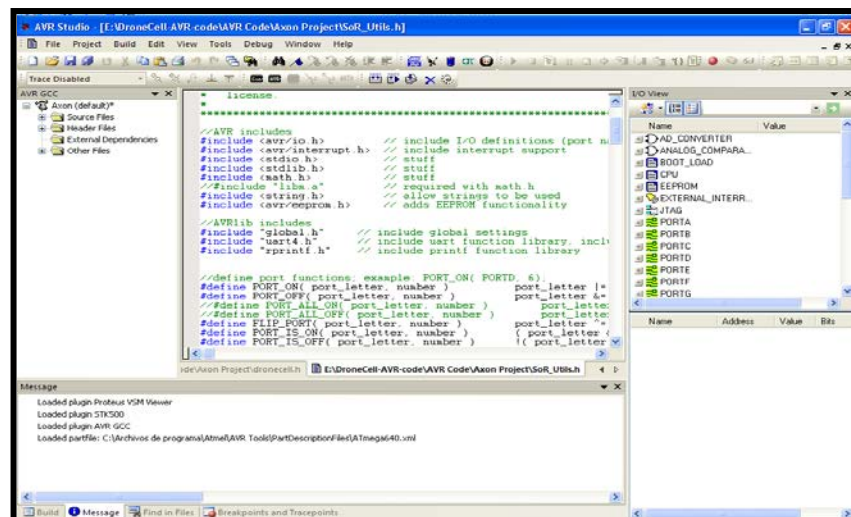


FIGURA 2-1: ENTORNO GRÁFICO DEL AVR STUDIO

El AVR STUDIO es una herramienta muy poderosa hecha por ATMEL la cual está orientada para los microcontroladores de la serie AT90S y para los AVR.

Esta herramienta nos permite escribir y depurar aplicaciones AVR, nos provee una herramienta para administrar proyectos, editores de texto, simuladores y es compatible con códigos como ASSEMBLER, C/C++ gracias a que tienen incorporado el compilador GNU/GCC.

El AVR STUDIO está compuesto por una ventana principal la cual es de mucha ayuda ya que nos da información sobre el control de flujo del programa (código), junto a la ventana principal hay otras ventanas las cuales nos permite tener un control total del estado de cada elemento en ejecución.

Las ventanas que acompañan a la ventana principal son:

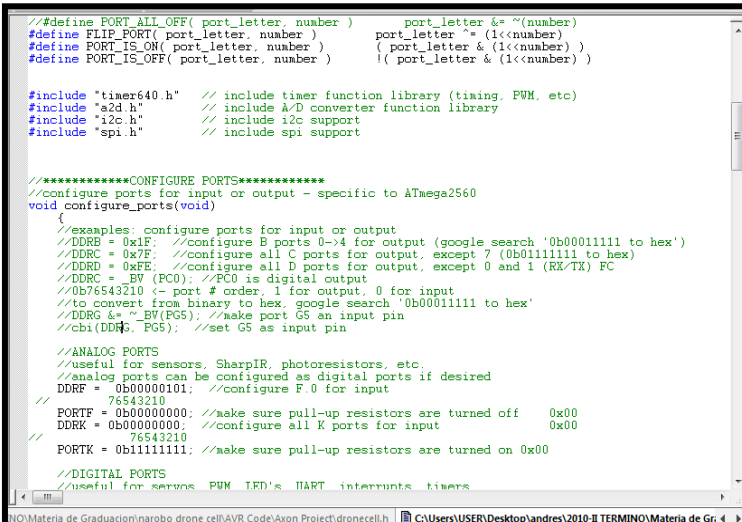
- Ventana de Registros.
- Ventana de Memorias.
- Ventana de Periféricos.
- Ventana de Mensajes.
- Ventana del Procesador.



### 2.1.1.1 Ventana Principal

Esta ventana se crea cuando un “object file” es creado o abierto y se encuentra presente durante toda la sesión, si esta ventana es cerrada, entonces la sesión se termina.

Como se indicó anteriormente, en ésta ventana se muestra el código que está siendo ejecutado (ver FIGURA 2-2).



```

//define PORT_ALL_OFF( port_letter, number )      port_letter &~ (1<<number)
#define FLIP_PORT( port_letter, number )         port_letter ^*(1<<number)
#define PORT_IS_ON( port_letter, number )        ( port_letter & (1<<number) )
#define PORT_IS_OFF( port_letter, number )        !( port_letter & (1<<number) )

#include "timer640.h" // include timer function library (timing, PWM, etc)
#include "a2d.h" // include A/D converter function library
#include "i2c.h" // include i2c support
#include "spi.h" // include spi support

//*****CONFIGURE PORTS*****
//configure ports for input or output - specific to ATmega2560
void configure_ports(void)
{
  //examples: configure ports for input or output
  //DDRB = 0x1F; //configure B ports 0-4 for output (google search '0b00011111 to hex')
  //DDRC = 0x7F; //configure all C ports for output, except 7 (0b01111111 to hex)
  //DDRD = 0xFF; //configure all D ports for output, except 0 and 1 (RX/TX) FC
  //DDRC = BV(PC0); //PC0 is digital output
  //0b76543210 <- port # order, 1 for output, 0 for input
  //to convert from binary to hex, google search '0b00011111 to hex'
  //DDRG &= ~BV(PG5); //make port G5 an input pin
  //cbi(DDH5, PG5); //set G5 as input pin

  //ANALOG PORTS
  //useful for sensors, SharpIR, photoresistors, etc.
  //analog ports can be configured as digital ports if desired
  //DDRF = 0b00000101; //configure F.0 for input
  //76543210
  PORTF = 0b00000000; //make sure pull-up resistors are turned off 0x00
  DDRK = 0b00000000; //configure all K ports for input 0x00
  //76543210
  PORTK = 0b11111111; //make sure pull-up resistors are turned on 0x00

  //DIGITAL PORTS
  //useful for servos, PWM, I2D's, UART, interrupts, timers

```

FIGURA 2-2: VENTANA PRINCIPAL DEL AVR STUDIO

En ésta ventana podemos usar herramientas que nos ayudan a tener mejor control sobre la ejecución del código como por ejemplo los “breakpoints”. Un breakpoint es un identificador que se ubica en la parte izquierda del programa y está representado por un punto rojo, además se puede poner un número ilimitado de breakpoints.

### 2.1.1.2 *Ventana de Registros*

Esta ventana nos muestra el contenido de los 32 registros del AVR.

Los valores de los registros pueden ser cambiados manualmente pero solo cuando la ejecución está detenida (ver FIGURA 2-3).

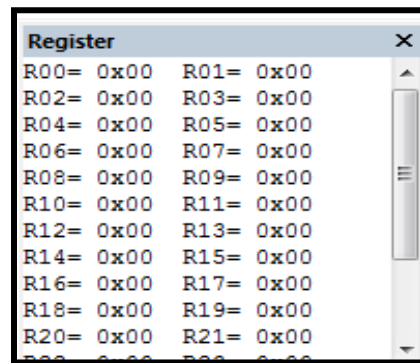


FIGURA 2-3: VENTANA DE REGISTROS

### 2.1.1.3 *Ventana de Memoria*

Esta ventana nos muestra el contenido de la memoria durante la ejecución del programa y también nos permite modificar los valores (ver FIGURA 2-4). Esta ventana además de mostrarnos la memoria del programa nos permite observar todo tipo de memoria (EEPROM, IO PORTS, REGISTROS).

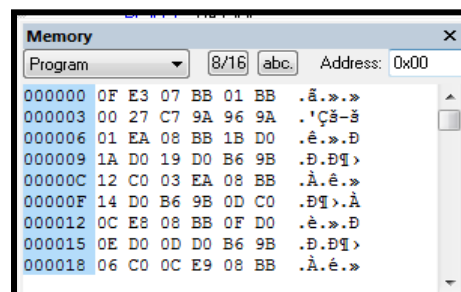


FIGURA 2-4: VENTANA DE MEMORIA

#### **2.1.1.4 Ventana de Mensajes.**

Muestra los mensajes del AVR STUDIO al usuario, cuando se presiona el botón de reset, todos los mensajes desaparecen ya que la ventana de mensajes es limpiada.

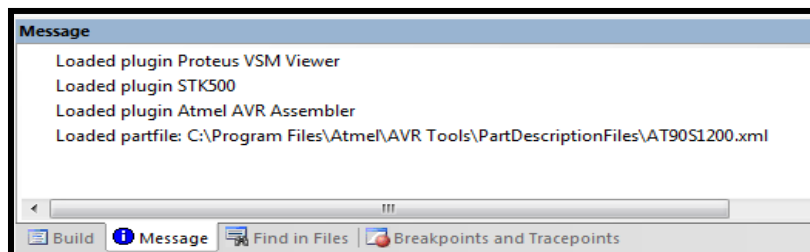


FIGURA 2-5: VENTANA DE MENSAJES

#### **2.1.1.5 Características del AVR STUDIO**

- La característica principal del AVR STUDIO es su fácil manejo.
- Permite visualizar rápidamente lo que está ocurriendo con los registros.
- Compila ASSEMBLER y tiene incorporado el compilador C (GNU/GCC).
- Realiza simulaciones por software.
- Es un software libre.

## 2.1.2 PROTEUS 7.7

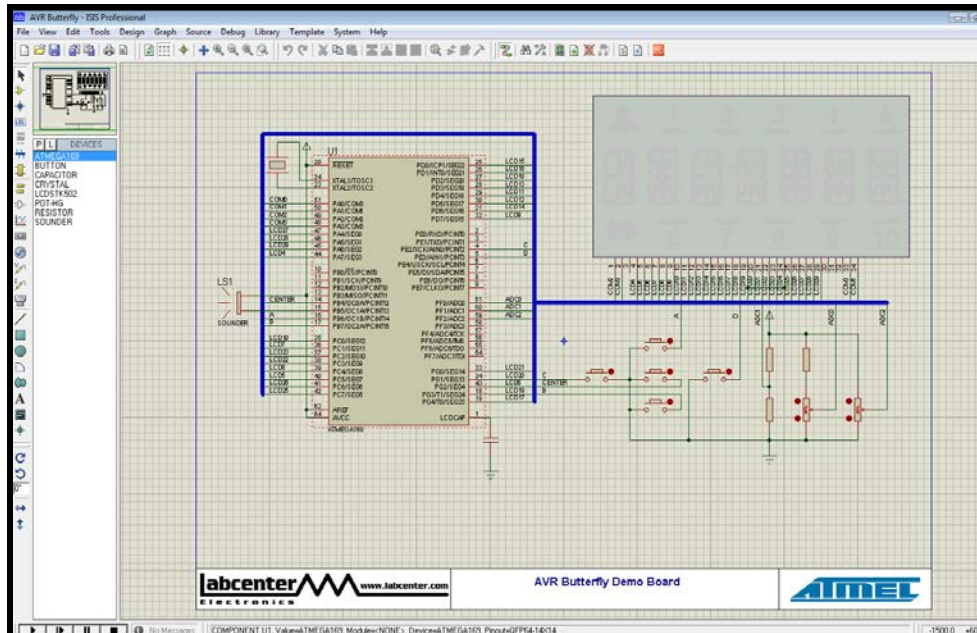


FIGURA 2-6: ENTORNO GRÁFICO DE PROTEUS V.7.7

Proteus es un software desarrollado por Labcenter Electronics y está orientado para la realización de proyectos electrónicos, éste programa nos permite diseñar y simular los diferentes circuitos electrónicos. El Proteus consta de dos partes importantes la cual es el ISIS y el ARES, los cuales nos van a permitir simular y diseñar las pistas para poder crear la placa física donde van a estar montados los elementos.

### **2.1.2.1**                    ***ISIS.- Intelligent Schematic Input System***

Esta herramienta es la que nos permite dibujar sobre un área de trabajo el circuito que posteriormente se procederá a simular, tiene incorporado una librería de más de 6.000 modelos de dispositivos digitales y analógicos como por ejemplo circuitos integrados, generadores de señales, resistencias, LCD, etc.

### **2.1.2.2**                    ***ARES. - Advanced Routing and Editing Software***

Esta herramienta sirve para enrutar, ubicar y editar los componentes del circuito, se utiliza para la fabricación de placas de circuito impreso (PCB's); además nos permite visualizarlas en 3D, de ésta manera podemos visualizar mejor el resultado final de la placa.

## 2.2 Herramientas de Hardware

### 2.2.1 POLOLU 3π

Es un Robot de alto rendimiento originalmente diseñado para la resolución de laberintos y como seguidor de línea pero en nuestro proyecto vamos a controlar los movimientos a través de mensajes de texto enviados desde un celular.

El corazón del Pololu 3π es el microcontrolador ATMEGA328P, éste microcontrolador tiene 32 KB flash, 2 KB RAM, y 1 KB de EEPROM, mas adelante detallaremos las características del ATmega328P (Ver ANEXO 2).

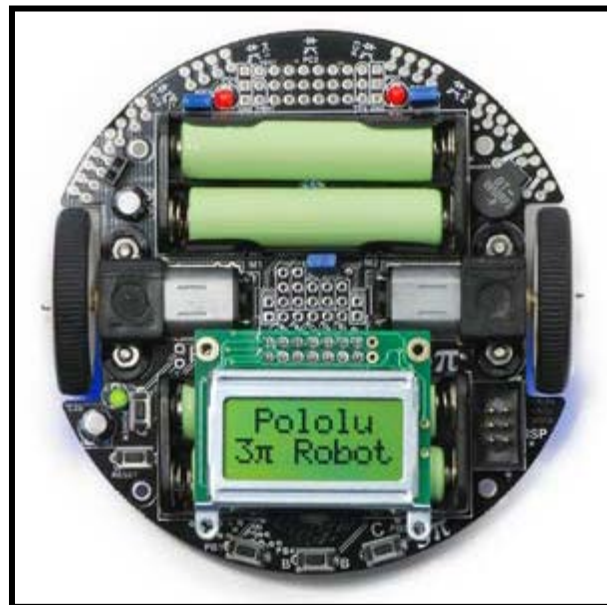


FIGURA 2-7: POLOLU 3π

El Pololu 3 $\pi$  tiene las siguientes características:

- AVR Conector ISP programador de 6-pin.
- Procesador ATmega328P
- Voltaje mínimo 3V
- Voltaje máximo 7V
- Máxima frecuencia de la señal PWM 80kHz
- Funciona perfectamente con el compilador GNU/GCC.
- ARV Studio ha desarrollado un ambiente de trabajo confortable.

En resumen el Robot seguidor de línea Pololu 3 $\pi$  está formado por sensores infrarrojos los cuales sirven de ayuda para que realice la función de seguidor, la programación del ATmega328 la podemos hacer a través del puerto ISP que viene incorporado en la placa del Pololu.

Este Robot opera con voltajes mínimos y máximos de 3 y 7 voltios respectivamente, es capaz de generar señales PWM.

### **2.2.1.1                    *ATmega328P***

Las características principales de este microcontrolador son:

- Posee 3 puertos (Port B, C, D).
- Cada puerto posee resistencias de Pull-up integradas.
- Tiene 3 timers (timer 0,1,2)
- Timer 0 y 2 son de 8 bits y el Timer 1 de 16bits.
- Tiene un comparador incorporado.
- Posee 14 pines donde 6 de ellos generan PWM.
- Genera señales PWM en los pines 3, 5, 6, 9, 10 y 11.

- Transmisión serial:
- Comunicación full-dúplex.
- Alta velocidad de transmisión, inclusive con cristales de baja frecuencia.
- Bit de paridad par o impar.
- Detección de falso bit de inicio.
- Detección de carácter perdido.
- Detección de error en el formato del carácter.
- Maneja niveles de voltaje TTL.
- 5, 6, 7,8 ó 9 bits de datos y 1 ó 2 bits de parada.
- Posee tres tipos de interrupciones diferentes: Transmisión Completa, Recepción Completa, Registro de transmisión vacío.
- Utiliza los pines 2 y 3 para Transmisión y Recepción respectivamente.
- Interrupciones externas en los pines 2 y 3
- Posee 32 registros de trabajo de propósito general y 23 líneas I/O de propósito general (Ver Anexo 4).



Es decir gracias a este microcontrolador podremos usar al Pololu 3π para realizar comunicaciones seriales con el otro dispositivo que vamos a utilizar el cual es el Narobo DroneCell.

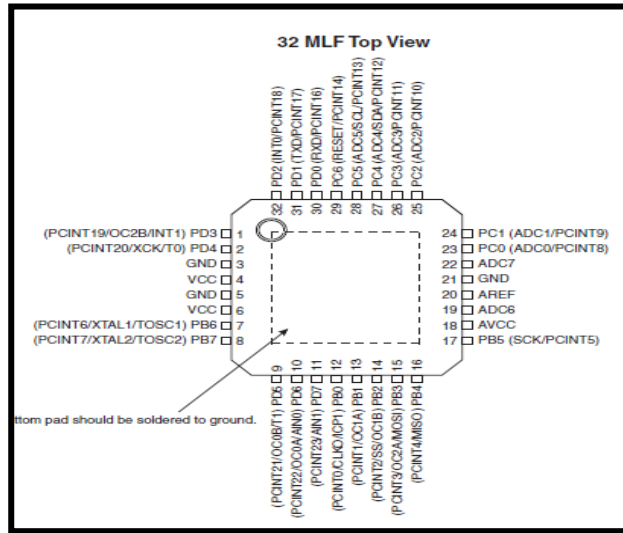


FIGURA 2-8: ATmega328P

En la siguiente figura se muestra una tabla en la cual mostramos las diferencias en memoria y el tamaño del vector de interrupción que existe entre el ATMEGA328P y otros ATMEGA (Ver ANEXO 1).

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 instruction words/vector

FIGURA 2-9: Tabla comparadora del ATmega328P con otros ATmegas

### **2.2.2 NAROBO DRONECELL**

La tarjeta Narobo DroneCell es una especie de celular para nuestros proyectos de robótica o electrónicos, éste dispositivo es comúnmente conocido como “todo en uno” en lo que respecta a comunicación ya que se puede comunicar vía mensajes de texto, se pueden realizar llamadas o incluso se puede comunicar hacia internet.

Otro punto importante de esta tarjeta es que cualquier dispositivo con TTL UART podrá comunicarse con la Tarjeta DroneCell. Con ésta tarjeta podemos realizar varias actividades como por ejemplo controlar dispositivos remotamente vía internet.

En nuestro proyecto desempeña un papel muy importante ya que es el que se va a encargar de recibir los mensajes de texto enviados desde nuestro celular y lo transmitirá al Pololu 3 $\pi$  utilizando la comunicación serial.

La FIGURA 2-11 muestra el nombre de los pines importantes de la tarjeta Narobo DroneCell.

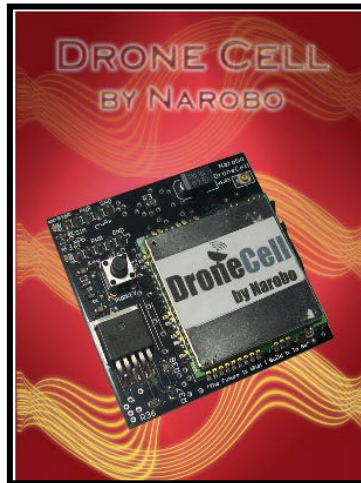


FIGURA 2-10: Narobo DroneCell

### **2.2.2.1 Características de la tarjeta Narobo DroneCell**

- Posee un led el cual indica el estado de la tarjeta.
- La interfaz UART trabaja con voltajes entre 3.3 V y 5V (mínimo y máximo respectivamente).
- EL voltaje de entrada de la fuente de poder está entre 5 - 16Vdc.
- El voltaje de regulación es de 4V.
- Posee una alta tasa de transmisión serial.
- La tasa de la Comunicación GPRS es de 86.5Kbps de bajada.
- Tasa de comunicación CSD es de 14.4Kbps.
- Trabaja con cualquier chip (SIM CARD).
- Posee un switch interno para detectar la presencia de alguna SIM CARD.
- Marca y recibe llamadas.
- Envía y recibe mensajes de texto.

- Envía y recibe datos desde cualquier computadora que tenga acceso a internet.
- Se pueden configurar alarmas.

Como se pudo apreciar en las características técnicas del Narobo, ésta cuenta con muchas herramientas útiles para la resolución de algunos proyectos de electrónica en el cual se involucre o se necesite alguna de estas cualidades que posee dicha tarjeta de comunicaciones.

En nuestro proyecto utilizaremos la característica de enviar y recibir mensajes de texto junto con la capacidad de transmitir serialmente a una velocidad de 115200 baudios.

Para poder utilizar la comunicación serial del Narobo, éste debe de tener un voltaje mínimo de 3.3V TTL y máximo de 5V TTL.

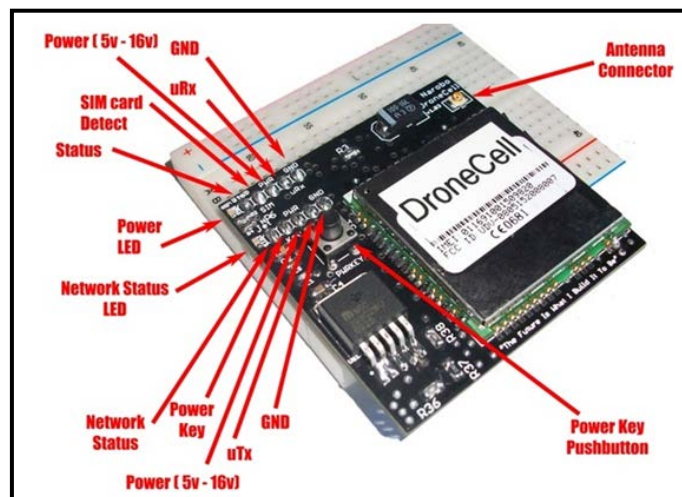


FIGURA 2-11: Pines del Narobo DroneCell

## **2.3 Transmisión de Mensajes vía celular**

### **2.3.1 Teléfonos GSM**

El lenguaje AT es usado por los teléfonos GSM para comunicarse con sus terminales, los comandos AT se utilizan para configurar y proporcionar instrucciones a los terminales, dichos comandos permiten realizar llamadas de datos o voz, leer y escribir en la agenda de contactos y enviar mensajes.

### **2.3.2 Lenguaje de Comunicación**

Los comandos AT son instrucciones codificadas que en principio fueron creadas para comunicar las terminales de los módems pero con el desarrollo de la tecnología ha aumentado, estos comandos fueron acondicionados para ser usados por GSM (Ver ANEXO 16,17).

#### **2.3.2.1 Comandos**

Comandos Generales

*AT+CGMI: Identificación del fabricante*

*AT+CGSN: Obtener número de serie*

*AT+CIMI: Obtener el IMSI*

*AT+CPAS: Leer estado del modem*

Comandos del servicio de red

*AT+CSQ: Obtener calidad de la señal*

*AT+COPS: Selección de un Operador*

*AT+CREG: Registrarse en una Red*

*AT+WOPN: Leer nombre del operador*

Comandos de seguridad

*AT+CPIN: Introducir el PIN*

*AT+CPINC: Obtener el número de reintentos que quedan*

*AT+CPWD: Cambiar password*

Comandos para la agenda de teléfonos

*AT+CPBR: Leer todas las entradas*

*AT+CPBF: Encontrar una entrada*

*AT+CPBW: Almacenar una entrada*

Comandos para SMS

*AT+CPMS: Seleccionar el lugar de almacenamiento de los SMS*

*AT+CMGF: Seleccionar el formato de los mensajes*

*AT+CMGR: Leer un mensaje SMS almacenado*

*AT+CMGL: Listar los mensajes almacenados*

*AT+CMGS: Enviar mensajes SMS*

*AT+CMGW: Almacenar mensaje en memoria*

*AT+CMSS: Enviar mensaje almacenado*

*AT+CSCA: Establecer el centro de mensajes a usar*

*AT+WMSC: Modificar el estado de un mensaje*

# CAPÍTULO 3

## DESCRIPCIÓN E IMPLEMENTACIÓN DEL PROYECTO

### 3.1 Diseño Preliminar

En los capítulos anteriores se había hecho una breve reseña sobre nuestro proyecto pero en éste capítulo se va a detallar las partes o etapas que lo conforman, las cuales las hemos dividido en tres partes fundamentales para el desarrollo del proyecto.

En dichas etapas se detallan como va interactuar el Pololu 3π con el Narobo DroneCell, para ello utilizamos un chip GSM conectado al Narobo DroneCell el cual se conecta a la red al momento de presionar el botón Power Key.

La señal transmitida desde un celular es recibida por el Narobo DroneCell, ésta señal será un mensaje de texto en el cual indicará las funciones o movimientos que debe realizar el Pololu 3π.

A continuación se detallan las etapas del Proyecto:

### 3.1.1 Comunicación entre Celular y Narobo DroneCell

Está compuesto por un Celular y el Narobo DroneCell.

Por medio del celular se procederá a enviar un mensaje de texto, el cual lo recibirá el chip GSM ubicado en la tarjeta Narobo DroneCell (previamente probado que la tarjeta se encuentra conectada a la red).

En esta comunicación, el mensaje que es transmitido por medio del celular hacia el Narobo, tendrá incluido las respectivas instrucciones que indicaran los movimientos al Pololu 3π.

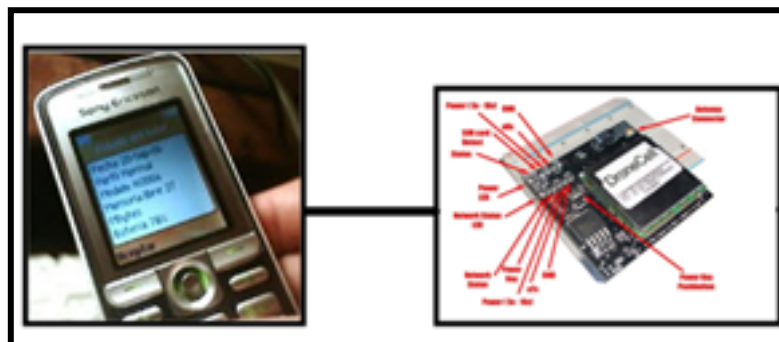


FIGURA 3-1: Componentes de la primera etapa del proyecto



### 3.1.2 Transmisión de Datos Serial

Está compuesto por el Narobo DroneCell y el Robot Pololu 3 $\pi$ . El DroneCell una vez que haya recibido el mensaje de texto procederá a transmitir mediante transmisión serial al Pololu 3 $\pi$ , dando paso a la última etapa.

Esta transmisión se lleva a cabo a través del pin 0 y 1 del puerto D del ATmega328P (Pololu 3 $\pi$ ), los cuales son receptor y transmisor respectivamente, éstos se conectan directamente al  $\mu$ Rx y  $\mu$ Tx del Narobo los cuales representan al transmisor y receptor TTL del DroneCell.

En ésta etapa del proyecto se va transmitir serialmente a una velocidad de 115200 baudios, los baudios son la unidad que se emplea para señalar la cantidad de bits por segundos que se transmiten.

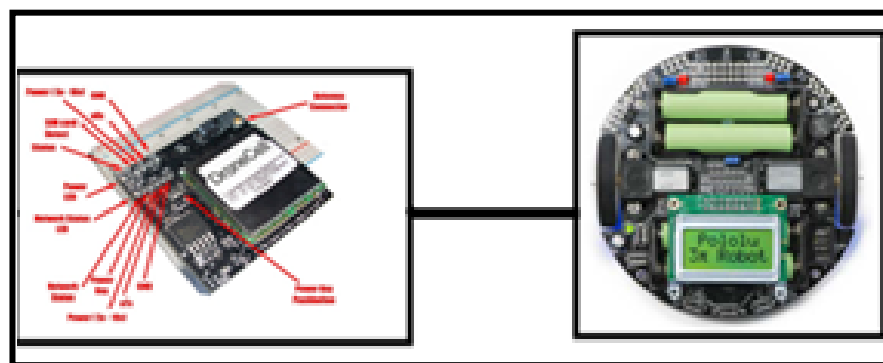


FIGURA 3-2: Componentes de la segunda etapa

### 3.1.3 Ejecución de los movimientos y formato del mensaje SMS

El Pololu 3π una vez que ha recibido el mensaje desde el Narobo DroneCell, procederá a revisar en su código y a comparar con las cadenas de texto predefinidas, de ésta manera se ejecutará la instrucción adecuada para efectuar el movimiento. A continuación se presenta el formato que se utiliza para validar las instrucciones.

- U ó u: Significa que el Pololu 3π se dirige hacia adelante.
- D ó d: Significa que el Pololu 3π se dirige hacia atrás.
- L ó l: Significa que el Pololu 3π se dirige hacia la izquierda.
- R ó r: Significa que el Pololu 3π se dirige hacia la derecha.

También el mensaje tiene un formato definido el cual se indica con un ejemplo U50L20D05R90, eso significa que el Pololu 3π se va a trasladar 50 segundos adelante, 20 segundos hacia la izquierda, luego 5 segundos hacia atrás y finalmente 90 segundos a la derecha. Cabe señalar que el mensaje puede tener más instrucciones de movimiento eso quiere decir que después de R90 se puede seguir con mas instrucciones y también el mensaje está validado para que se pueda recibir la instrucción con letras minúsculas, es decir el mensaje de ejemplo puede escribirse de la siguiente manera u50l20d05r90.

El numero después de la letra, tienen que ser de dos dígitos por ejemplo para enviar al Pololu que se mueva 5 segundos debe enviarse 05 y no solo 5. La velocidad del Pololu ya está predefinida.

A continuación se presenta las etapas del proyecto, ver FIGURA 3-3.

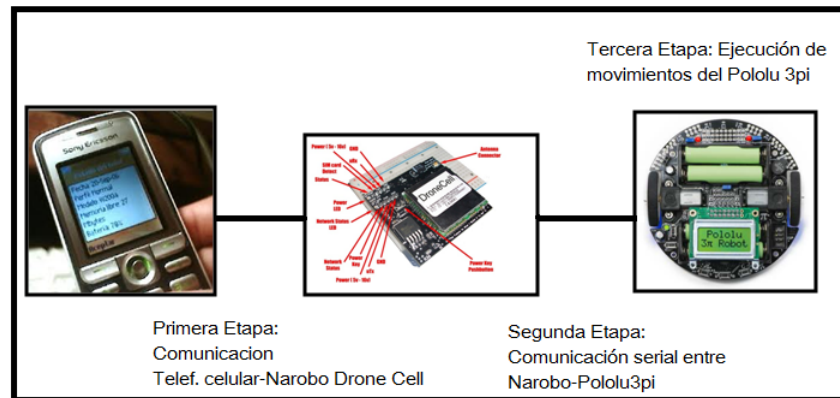


FIGURA 3-3: Componentes de las etapas del proyecto

### 3.1.4 Implementación Física

Para la implementación física de nuestro proyecto, utilizamos los componentes ya mencionados como son el Pololu 3 $\pi$ , la tarjeta Narobo DroneCell, chip GSM, teléfono celular, placa pcb perforada.

El Pololu 3 $\pi$  se alimenta con cuatro pilas AAA lo cual genera 5V, mientras que el Narobo DroneCell necesita de 5 – 16 V para funcionar, pero usamos las pilas AAA para alimentar todo el proyecto.

Adicional al DroneCell y al Pololu utilizamos una placa pcb la cual se va a encargar de realizar las conexiones de los dos módulos (Narobo DroneCell con el Pololu 3 $\pi$ ), éstas conexiones principalmente son para la fuente, la tierra, el Transmisor y el Receptor.

El pin de recepción del Pololu 3 $\pi$  (PD0) debe estar conectado con el pin  $\mu$ Rx del Narobo, y el pin de Transmisión del Pololu (PD1) se conecta con el pin  $\mu$ Tx del

Narobo DroneCell, los pines de fuente y tierra de ambos están conectados entre sí para de esa manera energizar ambas placas y hacer que compartan la tierra (GND).

También se usa una tarjeta SIM (chip GSM) para poder realizar la conexión con el teléfono celular, la cual se conecta al Narobo DroneCell.

## **3.2 Descripción del proyecto final**

### **3.2.1 Bienvenida**

Al momento de encender el Pololu 3π por medio de la botonera POWER se muestra un mensaje de bienvenida que es “Pololu 3π Welcome” este mensaje se mostrará en el display del Pololu por 5 segundos.

### **3.2.2 Establecer la comunicación**

Luego del mensaje de bienvenida aparecerá un nuevo mensaje “Press B to Start”, el cual indica que se debe presionar el botón B para comenzar con la comunicación, también se debe presionar la botonera del Narobo DroneCell para que el Pololu verifique el estado de la tarjeta Narobo. Si ésta funciona correcto se mostrará al final un mensaje en el display “Espero el SMS” que significa que el Narobo DroneCell está listo para recibir el mensaje de texto y ejecutar la instrucción indicada en el mensaje. Ver FIGURA 3-7.

### 3.3 Diagramas de Bloques del Proyecto

#### 3.3.1 Diagrama de bloques de la primera etapa

Mensaje enviado desde el teléfono celular y recibido por el Narobo gracias a la SIM que se le incorpora.

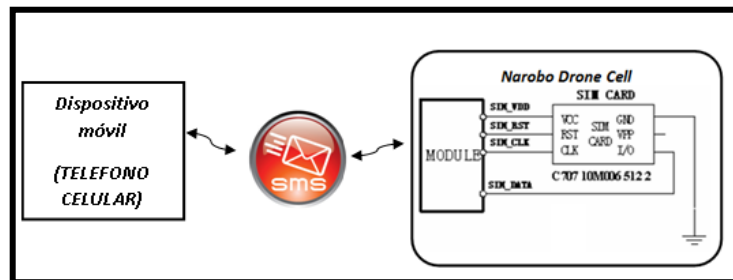


FIGURA 3-4: Diagrama de bloques primera etapa.

#### 3.3.2 Diagrama de bloques de la segunda etapa

En esta etapa el Narobo va a comunicarse con el Pololu 3π de manera serial, de tal manera que cualquier dato que reciba desde el celular va a ser enviado al Robot gracias a esta transmisión.

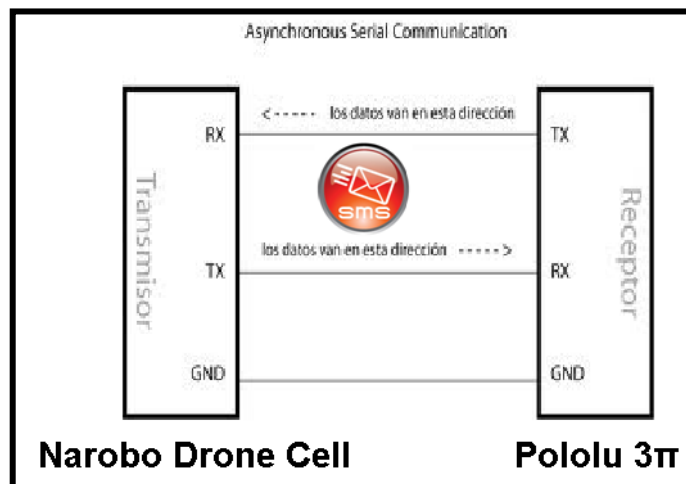


FIGURA 3-5: Diagrama de bloques segunda etapa.

### 3.3.3 Diagrama de bloques de la tercera etapa

El mensaje transmitido por el Narobo, es recibido por el Atmega328P (Pololu 3π).

En su interior se procesaran las cadenas de textos dando como resultado la instrucción del movimiento y como último paso, la ejecución de dicha instrucción.

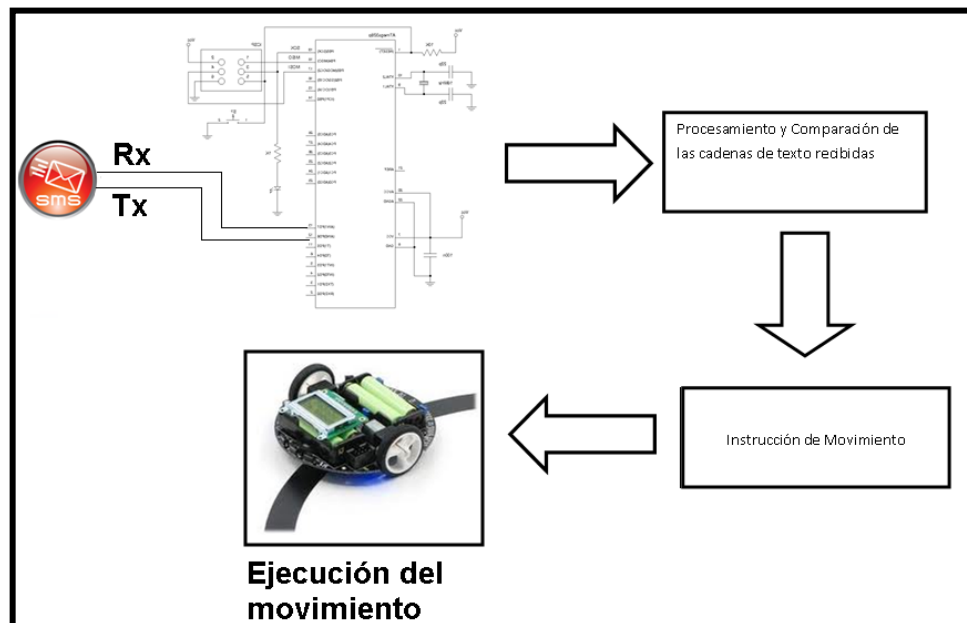


FIGURA 3-6: Diagrama de bloques tercera etapa.

### 3.3 Diagrama de flujo principal

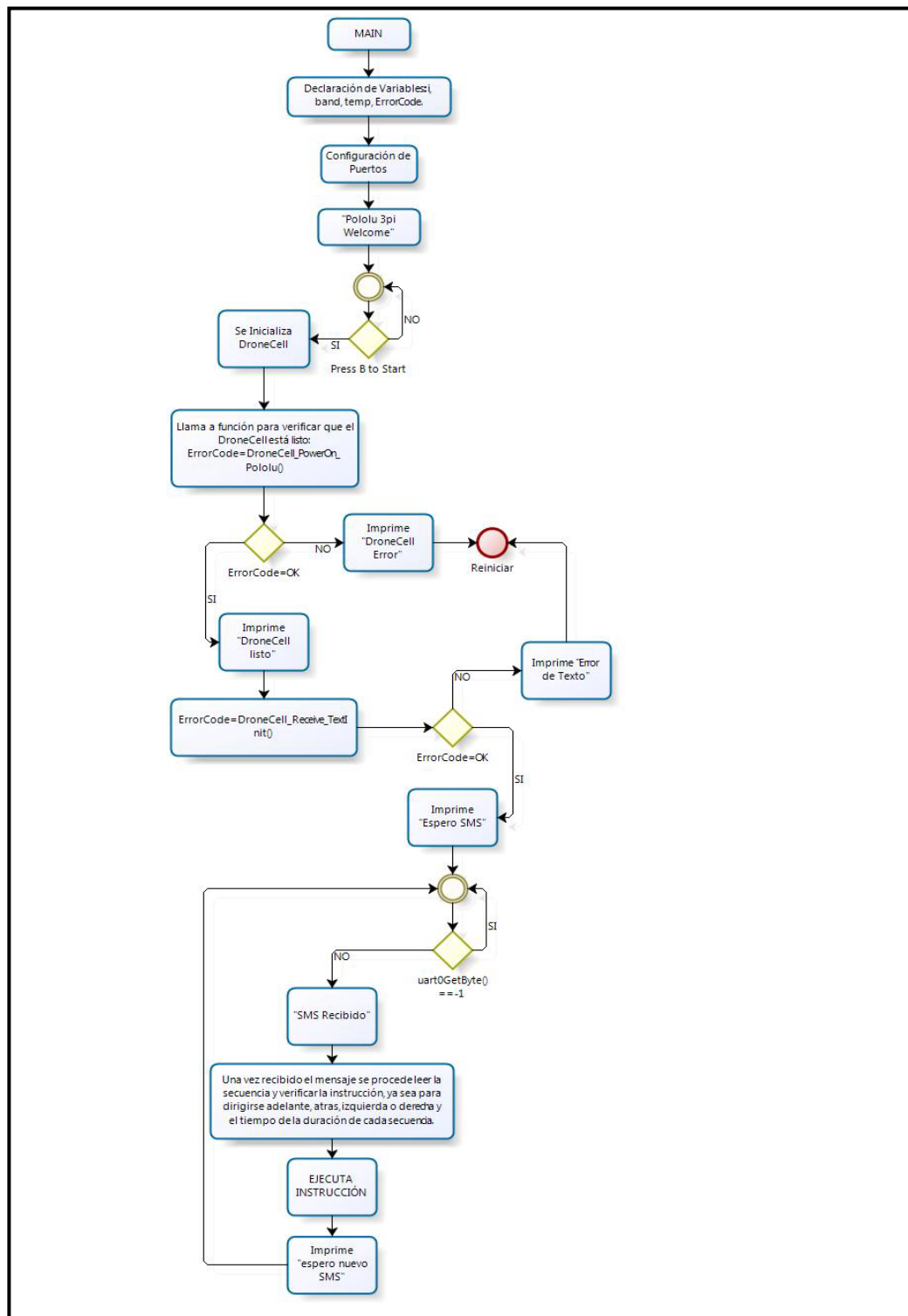


FIGURA 3-7: Diagrama de flujo del proyecto

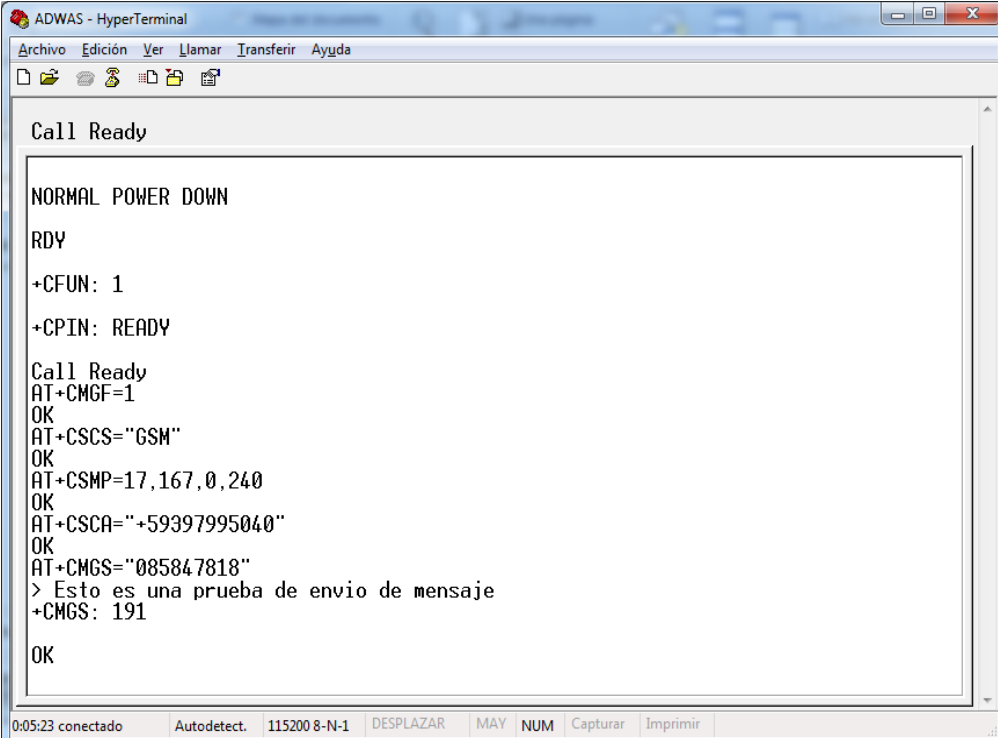
# CAPÍTULO 4

## SIMULACION Y PRUEBAS

### 4.1 Pruebas en Hyper Terminal

#### 4.1.1 Prueba de Transmisión

Para empezar a utilizar y verificar que el Narobo funciona realizamos pruebas utilizando el Hyper Terminal, a continuación se presenta el procedimiento para el envío de mensajes utilizando la tarjeta Narobo DroneCell ver FIGURA 4-1.



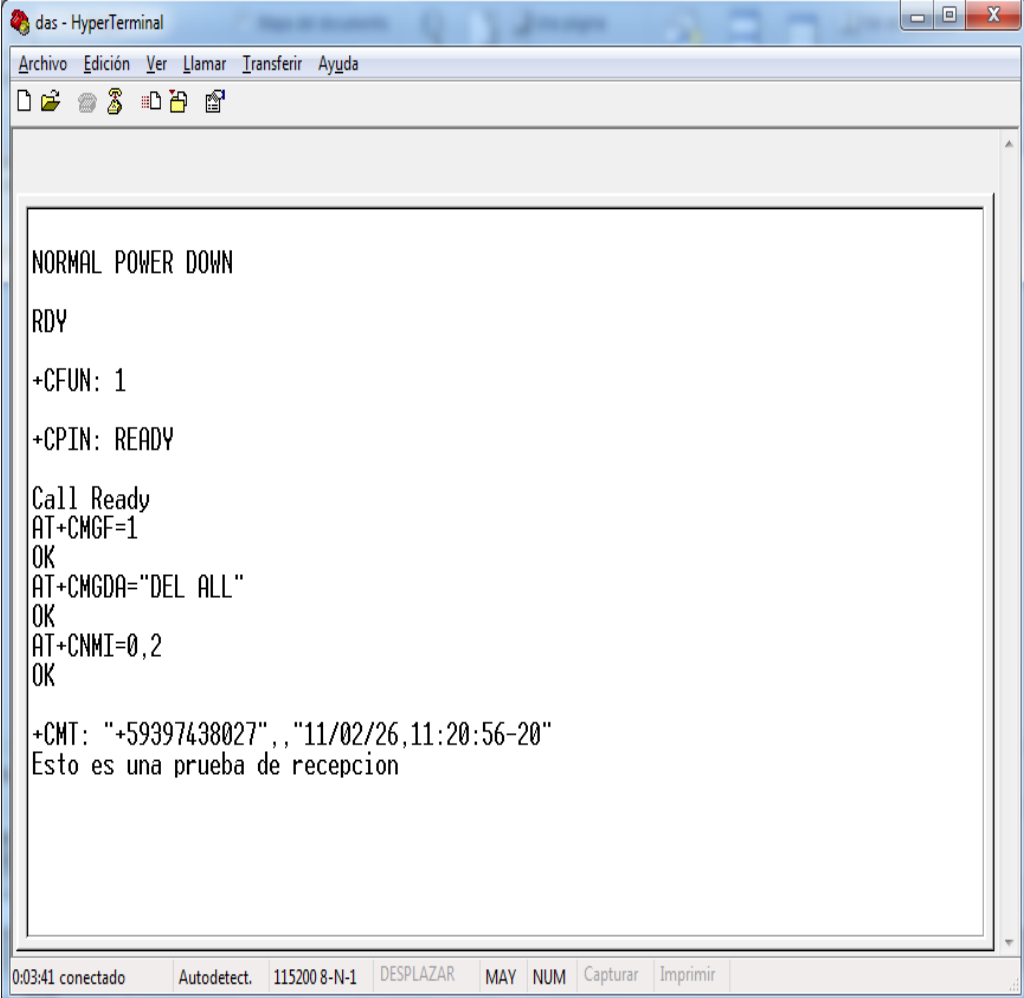
```
ADWAS - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda
Call Ready
NORMAL POWER DOWN
RDY
+CFUN: 1
+CPIN: READY
Call Ready
AT+CMGF=1
OK
AT+CSCS="GSM"
OK
AT+CSMP=17,167,0,240
OK
AT+CSCA="+59397995040"
OK
AT+CMGS="085847818"
> Esto es una prueba de envio de mensaje
+CMGS: 191
OK
0:05:23 conectado Autodetect. 115200 8-N-1 DESPLAZAR MAY NUM Capturar Imprimir
```

FIGURA 4-1 Prueba de Transmisión



### 4.1.2 Prueba de Recepción

También se realizó la prueba de recepción de mensajes, principalmente es la que utilizamos en nuestro proyecto, los siguientes pasos se deben seguir para recibir mensajes de texto enviados desde un celular.



```
das - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda
NORMAL POWER DOWN
RDY
+CFUN: 1
+CPIN: READY
Call Ready
AT+CMGF=1
OK
AT+CMGDA="DEL ALL"
OK
AT+CNMI=0,2
OK
+CMT: "+59397438027",,"11/02/26,11:20:56-20"
Esto es una prueba de recepcion
0:03:41 conectado Autodetect. 115200 8-N-1 DESPLAZAR MAY NUM Capturar Imprimir
```

FIGURA 4-2 Prueba de Recepción

## 4.2 Simulación en Proteus 7.7

Al iniciar la simulación en el programa Proteus 7.7, en el display del Pololu 3π sale un mensaje de bienvenida “Pololu 3π Welcome” dicho mensaje tiene una duración de 5 segundos ver FIGURA 4-3.

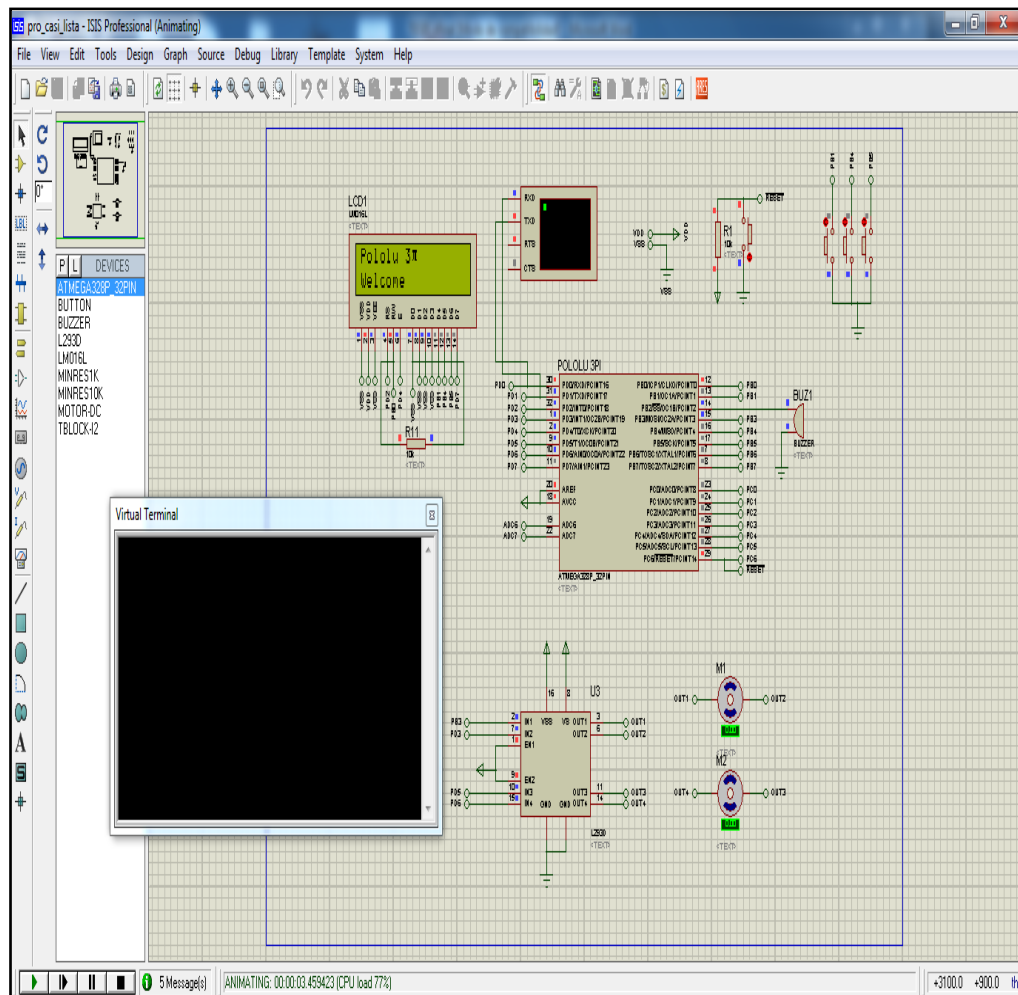


FIGURA 4-3 Iniciación del Proyecto Simulado

Luego del mensaje de bienvenida el Pololu presenta un mensaje “Press B to Start”, y se debe presionar el botón B para comenzar la comunicación con el Narobo ver FIGURA 4-4.

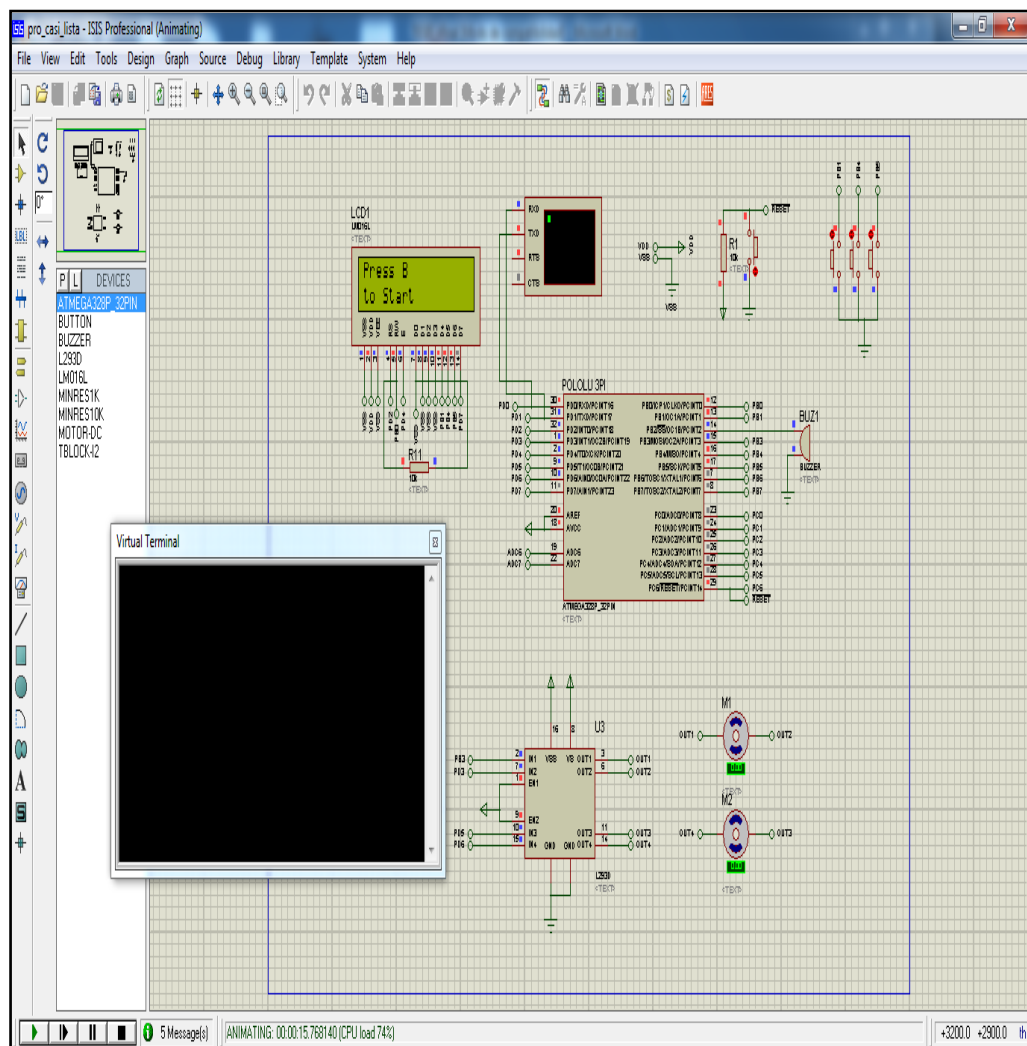


FIGURA 4-4 Press B to Start en Proteus

Al momento de Presionar el Botón B el Pololu 3π empieza la comunicación con la Tarjeta Narobo DroneCell, el Pololu presenta un mensaje indicando la Iniciación del DroneCell, lo cual significa que el Narobo se está iniciando y reconociendo la red FIGURA 4-5.

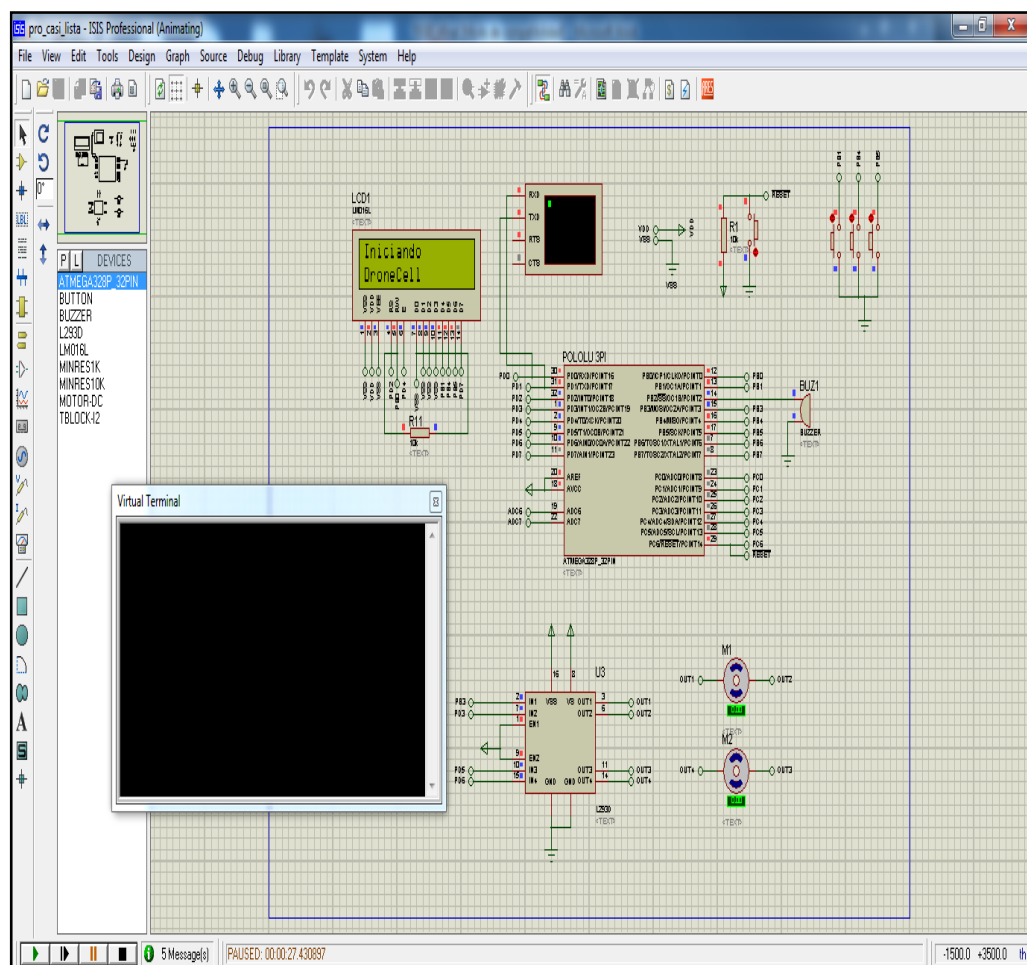


FIGURA 4-5 Iniciando DroneCell

Una vez iniciado el Narobo DroneCell el Pololu presenta un mensaje indicando que está Listo el DroneCell. En el Hyper Terminal presenta los códigos AT indicando la conexión lista del Narobo DroneCell en modo texto.

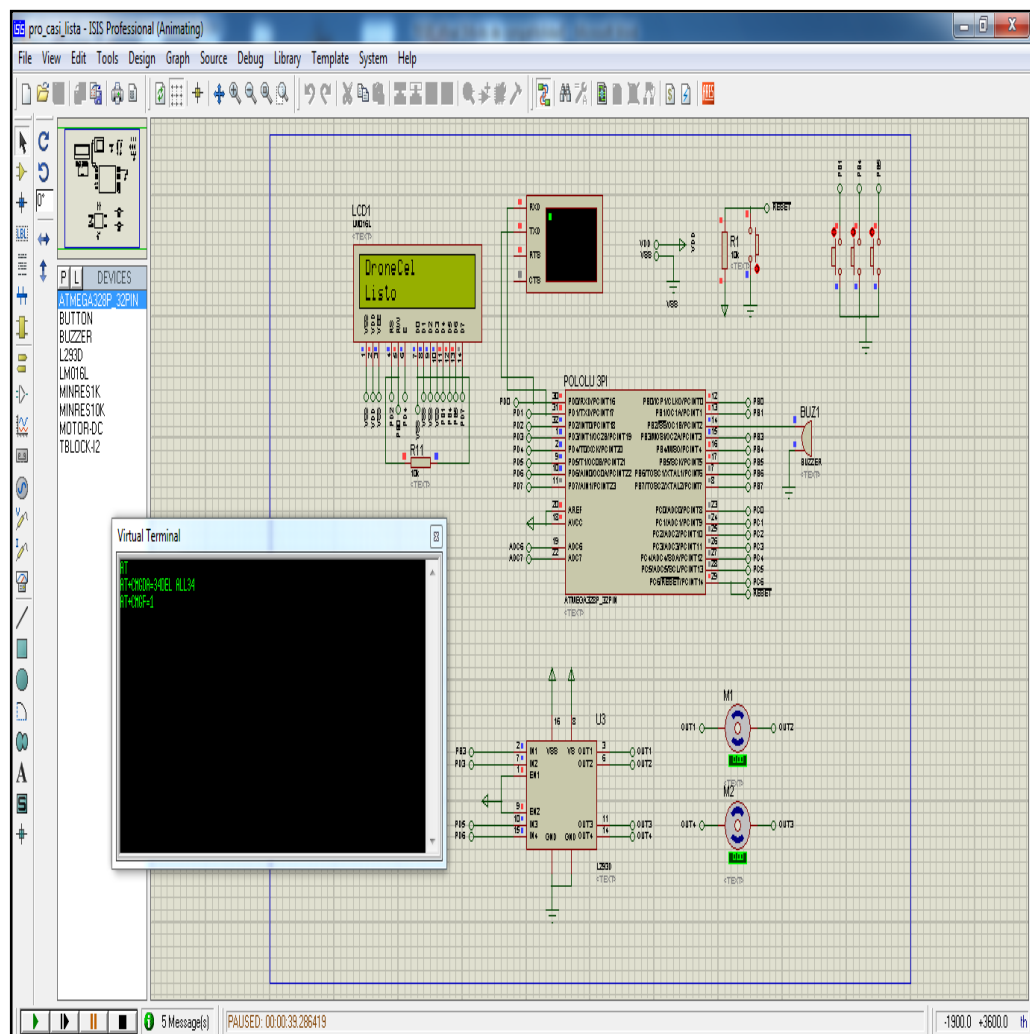


FIGURA 4-6 DroneCell Listo

En esta simulación el Pololu presenta un mensaje de error de texto ya que en Proteus no se puede realizar la comunicación inalámbrica, y en el código hay una función para verificar que exista comunicación inalámbrica y se reciba el código de mensaje de texto ver FIGURA 4-7.

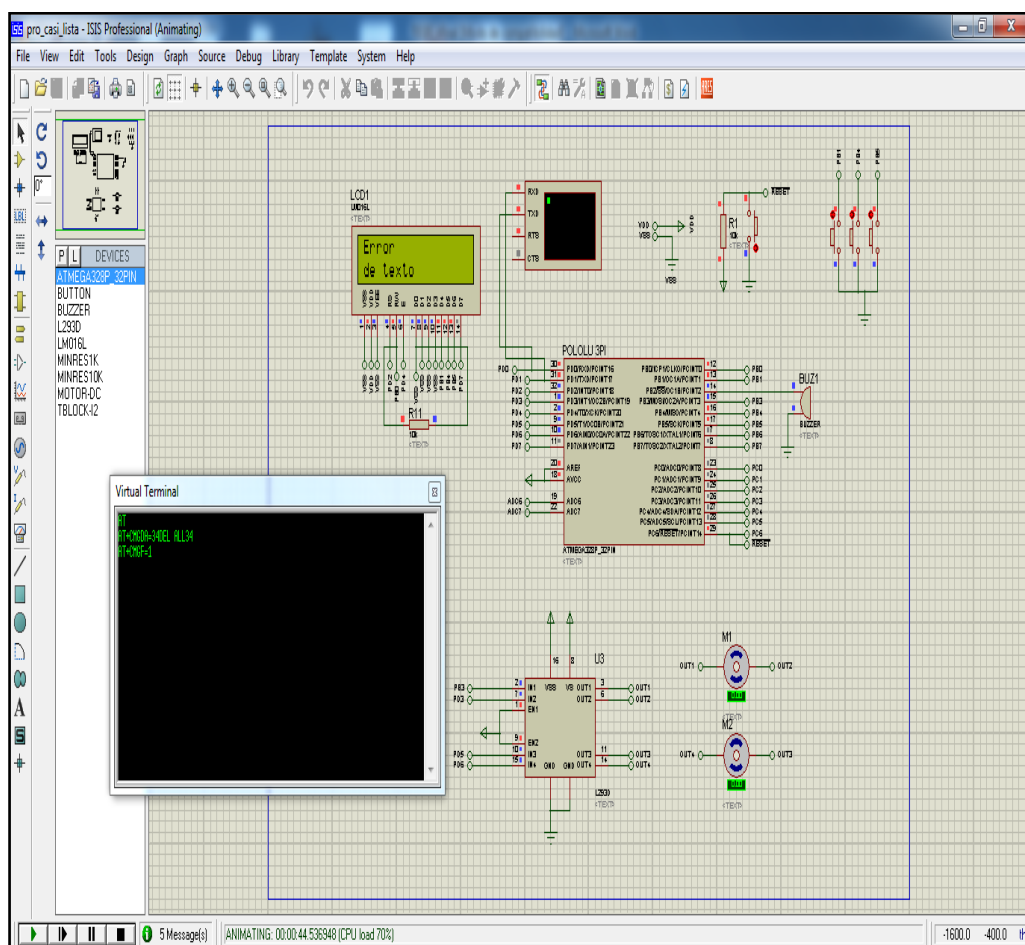


FIGURA 4-7 Error text

### 4.3 Resultados Experimentales

Los resultados de nuestro proyecto fueron exitosos, las siguientes figuras muestran las etapas en la que el Pololu presenta en la pantalla para que presione la tecla B y empezar con la Iniciación del DroneCell, luego verificar que esté listo y pueda recibir mensajes.

Luego de verificar la parte de mensaje, presentará el mensaje definitivo para que se envíe la instrucción por medio del mensaje de texto el cual es “Espero SMS”

A continuación se presentan los resultados experimentales del proyecto.

La FIGURA 4-8 muestra el mensaje para que se presione el botón B y pueda establecer la comunicación con el Narobo DroneCell.

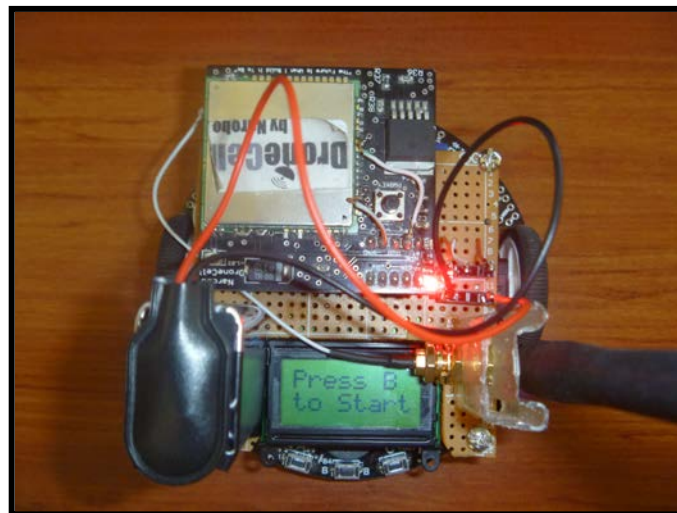


FIGURA 4-8 Press B to Start “experimental”

La FIGURA 4-9 presenta un mensaje indicando que espera el mensaje de texto para ejecutar la respectiva instrucción.

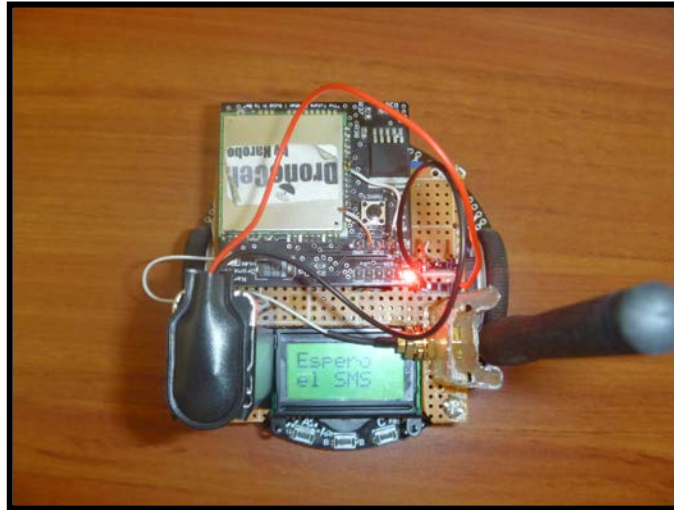


FIGURA 4-9 Espero el SMS

Cuando el mensaje llega al Narobo éste se transmite serialmente al Pololu para que se ejecute la respectiva instrucción.

Luego de ejecutar la Instrucción el Pololu presentará un mensaje que indica que está listo para recibir una nueva instrucción “Espero Nuevo SMS”.



FIGURA 4-10 Espero Nuevo SMS



# CONCLUSIONES

1. El corazón del Narobo DroneCell es un módulo para comunicaciones GSM de la serie SIM3XXDZ, el cual en éste proyecto vino previamente configurado con los respectivos comandos para entablar y realizar la comunicación serial con dispositivos que hablen el mismo lenguaje, es decir, está configurado para que reciba y transmita comandos AT los cuales son utilizados para ese tipo de comunicación.
2. Al realizar este proyecto se pudo establecer la gran utilidad que presenta el empleo de los celulares vinculados con microcontroladores para poder llevar a cabo diversas aplicaciones.
3. La tarjeta que utilizamos en interfaz con el Pololu 3π que es la Narobo DroneCell la cual presenta muchas ventajas ya que ésta tiene la facilidad de comunicación con el celular que es lo principal en nuestro proyecto para poder mover el Pololu por medio de un mensaje de texto con la instrucción específica.

4. Se puede concluir que el Robot Pololu 3π es un excelente dispositivo programable, el cual tiene incorporado el microcontrolador ATmega328P que es el que controla al Pololu 3π y tiene facilidad de uso para muchas aplicaciones.
5. También concluimos que las librerías de la tarjeta Narobo DroneCell son de mucha utilidad para la elaboración de códigos para transmitir datos enviados hacia el Narobo, de dichas librerías nos basamos para crear las funciones de recibir mensajes transmitidos desde un celular hacia el Narobo DroneCell y transmitirlos hacia el Pololu 3π.
6. La comunicación entre la tarjeta Narobo DroneCell y el Pololu 3π se la realizó fácilmente ya que el ATmega328P tiene el USART0 que transmite y recibe datos, la velocidad es programable, para nuestro trabajo utilizamos una frecuencia de 20MHz y una transmisión de 115200 baudios ya que el Narobo trabaja a esa velocidad de bits\seg.
7. La transmisión del mensaje de texto se la puede realizar desde cualquier celular que tenga conexión con red GSM, éste mensaje llega al Narobo DroneCell el cual tendrá un chip GSM para poder recibir el mensaje.

# RECOMENDACIONES

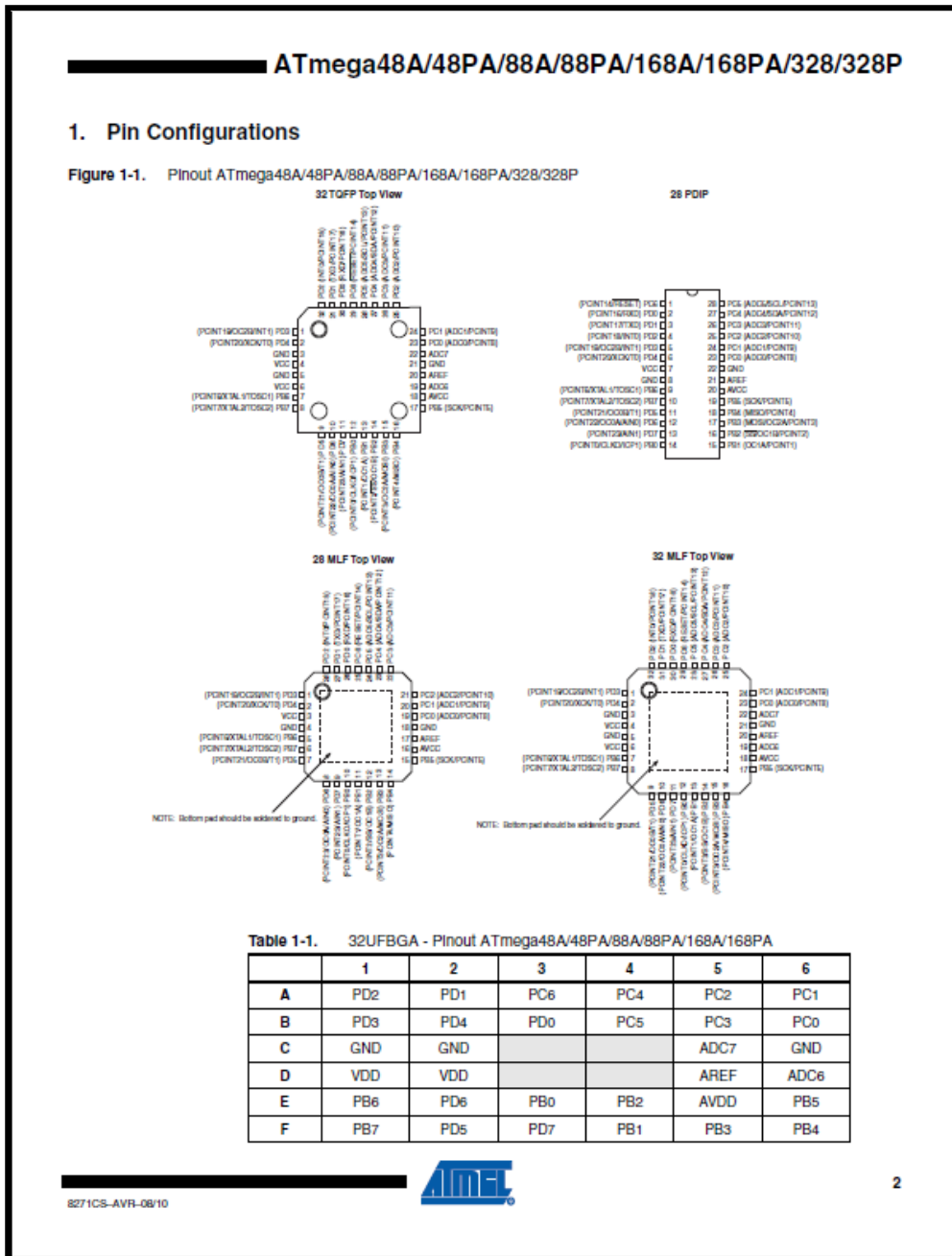
1. Para la primera etapa del proyecto, tenemos que asegurarnos que el Narobo DroneCell una vez que ha sido energizado (led power encendido) y presionado el botón "pwrkey" debe de tener el led de status activado (parpadeando) y se debe de haber confirmado que el chip GSM se encuentre activo (habilitado y conectado a la red GSM de su operador de telefonía celular).
2. Asegurarse que le llegue el voltaje adecuado a la tarjeta Narobo, el regulador de voltaje esté operando en zona lineal y con la ayuda de un osciloscopio observar los pines de transmisión y recepción de datos del UART para chequear que efectivamente se realice la transmisión en el momento indicado.
3. Una vez soldado todos los leds en el Pololu 3π, revisar que el voltaje y corriente entregado al Robot sea la suficiente para óptimas condiciones de operación, ya que si no se cumplen esas condiciones pueden afectar a otros elementos del Robot como el LCD que en nuestro proyecto al momento de soldar esos leds, comenzó a presentar basura en pantalla y en ocasiones no presentaba nada.

4. Se recomienda hacer simulaciones con el Hyper Terminal ya que tiene mucha utilidad para verificar que se están transmitiendo o recibiendo los datos correctos.
  
5. El Narobo DroneCell requiere su configuración para poder recibir mensajes lo cual se realiza por medio de códigos AT por lo tanto se recomienda entender los comandos AT para su configuración.
  
6. Como el Pololu 3π tiene un tamaño pequeño y la tarjeta Narobo DroneCell con su respectiva antena tienen que estar conectados, se recomienda hacer una placa e incorporarla con el Pololu 3π, para así poder realizar la conexión serial y también la antena quede en una posición fija.

# ANEXOS

## ANEXO 1

### Modelo Físico del ATmega328P



## ANEXO 2

### Hoja de datos del ATmega328P

**ATmega48A/48PA/88A/88PA/168A/168PA/328/328P**

#### 1.1 Pin Descriptions

**1.1.1 VCC**  
Digital supply voltage.

**1.1.2 GND**  
Ground.

**1.1.3 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2**  
Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the Inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the Inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7...6 is used as TOSC2...1 Input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in ["System Clock and Clock Options"](#) on page 26.


**1.1.4 Port C (PC5:0)**  
Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5...0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.


**1.1.5 PC6/RESET**  
If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset Input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in [Table 28-12 on page 323](#). Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in ["Alternate Functions of Port C"](#) on page 86.

**1.1.6 Port D (PD7:0)**  
Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

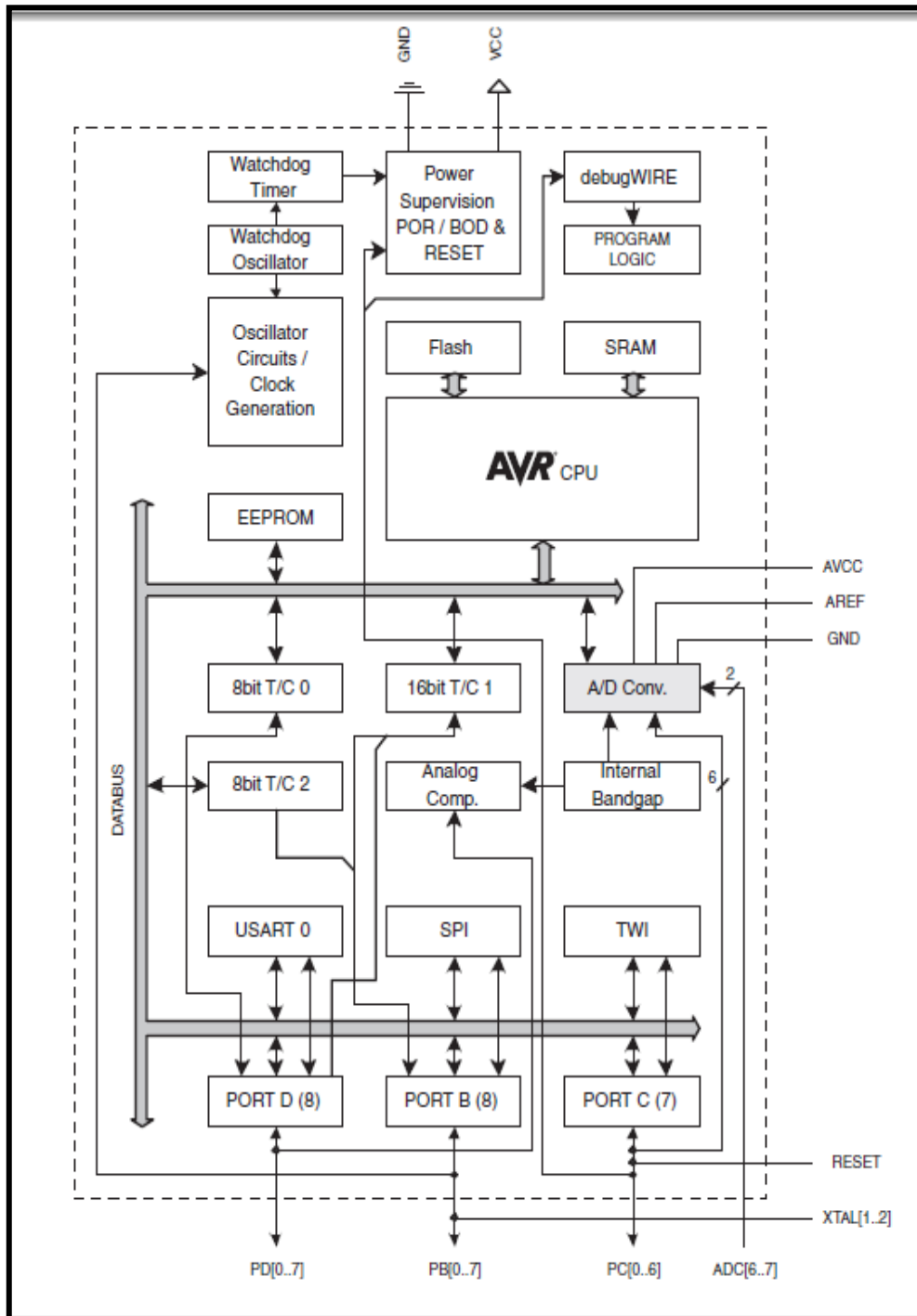
  
8271CS-AVR-08/10



3

### ANEXO 3

#### Arquitectura del ATmega328P



## ANEXO 4

### Registros del ATmega328P

ATmega48A/48PA/88A/88PA/168A/168PA/328/328P										
4. Register Summary										
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0xFF	Reserved	--	--	--	--	--	--	--	--	
0xFE	Reserved	--	--	--	--	--	--	--	--	
0xFD	Reserved	--	--	--	--	--	--	--	--	
0xFC	Reserved	--	--	--	--	--	--	--	--	
0xFB	Reserved	--	--	--	--	--	--	--	--	
0xFA	Reserved	--	--	--	--	--	--	--	--	
0xF9	Reserved	--	--	--	--	--	--	--	--	
0xF8	Reserved	--	--	--	--	--	--	--	--	
0xF7	Reserved	--	--	--	--	--	--	--	--	
0xF6	Reserved	--	--	--	--	--	--	--	--	
0xF5	Reserved	--	--	--	--	--	--	--	--	
0xF4	Reserved	--	--	--	--	--	--	--	--	
0xF3	Reserved	--	--	--	--	--	--	--	--	
0xF2	Reserved	--	--	--	--	--	--	--	--	
0xF1	Reserved	--	--	--	--	--	--	--	--	
0xF0	Reserved	--	--	--	--	--	--	--	--	
0xEF	Reserved	--	--	--	--	--	--	--	--	
0xEE	Reserved	--	--	--	--	--	--	--	--	
0xED	Reserved	--	--	--	--	--	--	--	--	
0xEC	Reserved	--	--	--	--	--	--	--	--	
0xEB	Reserved	--	--	--	--	--	--	--	--	
0xEA	Reserved	--	--	--	--	--	--	--	--	
0xE9	Reserved	--	--	--	--	--	--	--	--	
0xE8	Reserved	--	--	--	--	--	--	--	--	
0xE7	Reserved	--	--	--	--	--	--	--	--	
0xE6	Reserved	--	--	--	--	--	--	--	--	
0xE5	Reserved	--	--	--	--	--	--	--	--	
0xE4	Reserved	--	--	--	--	--	--	--	--	
0xE3	Reserved	--	--	--	--	--	--	--	--	
0xE2	Reserved	--	--	--	--	--	--	--	--	
0xE1	Reserved	--	--	--	--	--	--	--	--	
0xE0	Reserved	--	--	--	--	--	--	--	--	
0xDF	Reserved	--	--	--	--	--	--	--	--	
0xDE	Reserved	--	--	--	--	--	--	--	--	
0xDD	Reserved	--	--	--	--	--	--	--	--	
0xDC	Reserved	--	--	--	--	--	--	--	--	
0xDB	Reserved	--	--	--	--	--	--	--	--	
0xDA	Reserved	--	--	--	--	--	--	--	--	
0xD9	Reserved	--	--	--	--	--	--	--	--	
0xD8	Reserved	--	--	--	--	--	--	--	--	
0xD7	Reserved	--	--	--	--	--	--	--	--	
0xD6	Reserved	--	--	--	--	--	--	--	--	
0xD5	Reserved	--	--	--	--	--	--	--	--	
0xD4	Reserved	--	--	--	--	--	--	--	--	
0xD3	Reserved	--	--	--	--	--	--	--	--	
0xD2	Reserved	--	--	--	--	--	--	--	--	
0xD1	Reserved	--	--	--	--	--	--	--	--	
0xD0	Reserved	--	--	--	--	--	--	--	--	
0xCF	Reserved	--	--	--	--	--	--	--	--	
0xCE	Reserved	--	--	--	--	--	--	--	--	
0xCD	Reserved	--	--	--	--	--	--	--	--	
0xCC	Reserved	--	--	--	--	--	--	--	--	
0xCB	Reserved	--	--	--	--	--	--	--	--	
0xCA	Reserved	--	--	--	--	--	--	--	--	
0xC9	Reserved	--	--	--	--	--	--	--	--	
0xC8	Reserved	--	--	--	--	--	--	--	--	
0xC7	Reserved	--	--	--	--	--	--	--	--	
0xC6	LDRs									
0xC5	USART I/O Data Register									198
0xC4	USRRSH									200
0xC3	USRRSL									200
0xC2	Reserved									
0xC1	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSR0A/UCSR0B	UCSR0E/UCSR0F	UCPOL0	199/213



## ANEXO 5

### Registros del ATmega328P

ATmega48A/48PA/88A/88PA/168A/168PA/328/328P											
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page	
(0x01)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZA0	RXB00	TXB00	197	
(0x02)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	Uz0	MPCM0	196	
(0x0F)	Reserved	--	--	--	--	--	--	--	--		
(0x1E)	Reserved	--	--	--	--	--	--	--	--		
(0x20)	TWAMR	TWAM0	TWAM0	TWAM0	TWAM0	TWAM0	TWAM0	TWAM0	--	245	
(0x21)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	--	TWIE	242	
(0x22)	TWDR	2-wire Serial Interface Data Register									244
(0x2A)	TWAR	TWA0	TWA0	TWA0	TWA0	TWA0	TWA0	TWA0	TWGE	245	
(0x2B)	TWSR	TWS7	TWSe	TWSe	TWSe	TWSe	--	TWPS1	TWPS0	244	
(0x2E)	TWBR	2-wire Serial Interface Bit Rate Register									242
(0x37)	Reserved	--	--	--	--	--	--	--	--		
(0x38)	ASPR	--	EXCLK	ASP	TCN0UB	OCR0AUB	OCR0BUB	TCR0AUB	TCR0BUB	165	
(0x39)	Reserved	--	--	--	--	--	--	--	--		
(0x3A)	OCR0B	Timer/Counter0 Output Compare Register B									163
(0x3B)	OCR0A	Timer/Counter0 Output Compare Register A									163
(0x3C)	TCNT0	Timer/Counter0 (8-bit)									163
(0x3D)	TCCR0B	FOC0A	FOC0B	--	--	WGM02	CS02	CS01	CS00	162	
(0x3E)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	--	--	WGM01	WGM00	159	
(0x3F)	Reserved	--	--	--	--	--	--	--	--		
(0x4E)	Reserved	--	--	--	--	--	--	--	--		
(0x4D)	Reserved	--	--	--	--	--	--	--	--		
(0x4C)	Reserved	--	--	--	--	--	--	--	--		
(0x4B)	Reserved	--	--	--	--	--	--	--	--		
(0x4A)	Reserved	--	--	--	--	--	--	--	--		
(0x49)	Reserved	--	--	--	--	--	--	--	--		
(0x48)	Reserved	--	--	--	--	--	--	--	--		
(0x47)	Reserved	--	--	--	--	--	--	--	--		
(0x46)	Reserved	--	--	--	--	--	--	--	--		
(0x45)	Reserved	--	--	--	--	--	--	--	--		
(0x44)	Reserved	--	--	--	--	--	--	--	--		
(0x43)	Reserved	--	--	--	--	--	--	--	--		
(0x42)	Reserved	--	--	--	--	--	--	--	--		
(0x41)	Reserved	--	--	--	--	--	--	--	--		
(0x40)	Reserved	--	--	--	--	--	--	--	--		
(0x3F)	Reserved	--	--	--	--	--	--	--	--		
(0x3E)	Reserved	--	--	--	--	--	--	--	--		
(0x3D)	Reserved	--	--	--	--	--	--	--	--		
(0x3C)	Reserved	--	--	--	--	--	--	--	--		
(0x3B)	Reserved	--	--	--	--	--	--	--	--		
(0x3A)	Reserved	--	--	--	--	--	--	--	--		
(0x39)	Reserved	--	--	--	--	--	--	--	--		
(0x38)	Reserved	--	--	--	--	--	--	--	--		
(0x37)	Reserved	--	--	--	--	--	--	--	--		
(0x36)	Reserved	--	--	--	--	--	--	--	--		
(0x35)	Reserved	--	--	--	--	--	--	--	--		
(0x34)	Reserved	--	--	--	--	--	--	--	--		
(0x33)	Reserved	--	--	--	--	--	--	--	--		
(0x32)	Reserved	--	--	--	--	--	--	--	--		
(0x31)	Reserved	--	--	--	--	--	--	--	--		
(0x30)	Reserved	--	--	--	--	--	--	--	--		
(0x2F)	Reserved	--	--	--	--	--	--	--	--		
(0x2E)	Reserved	--	--	--	--	--	--	--	--		
(0x2D)	Reserved	--	--	--	--	--	--	--	--		
(0x2C)	Reserved	--	--	--	--	--	--	--	--		
(0x2B)	Reserved	--	--	--	--	--	--	--	--		
(0x2A)	Reserved	--	--	--	--	--	--	--	--		
(0x29)	Reserved	--	--	--	--	--	--	--	--		
(0x28)	Reserved	--	--	--	--	--	--	--	--		
(0x27)	Reserved	--	--	--	--	--	--	--	--		
(0x26)	Reserved	--	--	--	--	--	--	--	--		
(0x25)	Reserved	--	--	--	--	--	--	--	--		
(0x24)	Reserved	--	--	--	--	--	--	--	--		
(0x23)	Reserved	--	--	--	--	--	--	--	--		
(0x22)	Reserved	--	--	--	--	--	--	--	--		
(0x21)	Reserved	--	--	--	--	--	--	--	--		
(0x20)	Reserved	--	--	--	--	--	--	--	--		
(0x1F)	Reserved	--	--	--	--	--	--	--	--		
(0x1E)	Reserved	--	--	--	--	--	--	--	--		
(0x1D)	Reserved	--	--	--	--	--	--	--	--		
(0x1C)	Reserved	--	--	--	--	--	--	--	--		
(0x1B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte									139
(0x1A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte									139
(0x19)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte									139
(0x18)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte									139
(0x17)	ICR1H	Timer/Counter1 - Input Capture Register High Byte									139
(0x16)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte									139
(0x15)	TCNT1H	Timer/Counter1 - Counter Register High Byte									139
(0x14)	TCNT1L	Timer/Counter1 - Counter Register Low Byte									139
(0x13)	Reserved	--	--	--	--	--	--	--	--		
(0x12)	TCCR1C	FOC1A	FOC1B	--	--	--	--	--	--	138	
(0x11)	TCCR1B	ICNC1	ICES1	--	--	WGM13	WGM12	CS12	CS11	CS10	
(0x10)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	--	--	--	WGM11	WGM10	



# ANEXO 6

## Registros del ATmega328P

ATmega48A/48PA/88A/88PA/168A/168PA/328/328P										
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x0F	DIDR1	--	--	--	--	--	--	AIN1D	AIN0D	250
0x0E	DIDR0	--	--	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	267
0x0D	Reserved	--	--	--	--	--	--	--	--	
0x0C	ADMUX	REFS1	REFS0	ADLAR1	--	MUX3	MUX2	MUX1	MUX0	263
0x0B	ADCSRB	--	ACME	--	--	--	ADTS2	ADTS1	ADTS0	266
0x0A	ADCSRA	ADEF	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	264
0x09	ADCH	ADC Data Register High byte								266
0x08	ADCL	ADC Data Register Low byte								266
0x07	Reserved	--	--	--	--	--	--	--	--	
0x06	Reserved	--	--	--	--	--	--	--	--	
0x05	Reserved	--	--	--	--	--	--	--	--	
0x04	Reserved	--	--	--	--	--	--	--	--	
0x03	Reserved	--	--	--	--	--	--	--	--	
0x02	Reserved	--	--	--	--	--	--	--	--	
0x01	Reserved	--	--	--	--	--	--	--	--	
0x00	Reserved	--	--	--	--	--	--	--	--	
0x0F	TIMSK2	--	--	--	--	--	OCIE2B	OCIE2A	TOIE2	164
0x0E	TIMSK1	--	--	ICF1	--	--	OCIE1B	OCIE1A	TOIE1	140
0x0D	TIMSK0	--	--	--	--	--	OCIE0B	OCIE0A	TOIE0	112
0x0C	PCMSK2	PCINT2a	PCINT2z	PCINT2n	PCINT2o	PCINT1a	PCINT1b	PCINT17	PCINT16	75
0x0B	PCMSK1	--	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	75
0x0A	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	75
0x09	Reserved	--	--	--	--	--	--	--	--	
0x08	EICRA	--	--	--	--	ISC11	ISC10	ISC01	ISC00	72
0x07	EICRB	--	--	--	--	--	PCIE2	PCIE1	PCIE0	
0x06	Reserved	--	--	--	--	--	--	--	--	
0x05	OSCCAL	Oscillator Calibration Register								37
0x04	Reserved	--	--	--	--	--	--	--	--	
0x03	PRR	PRTW	PRTMz	PRTMo	--	PRTM1	PRSPI	PRUSART0	PRADC	42
0x02	Reserved	--	--	--	--	--	--	--	--	
0x01	Reserved	--	--	--	--	--	--	--	--	
0x00	Reserved	--	--	--	--	--	--	--	--	
0x0F	CLKPR	CLKPCE	--	--	--	CLKPS3	CLKPS2	CLKPS1	CLKPS0	37
0x0E	WDTCR	WDFR	WDIE	WDF3	WDF2	WDF1	WDF0	WDFP	WDFB	55
0x0D	SREG	I	T	H	S	V	N	Z	C	9
0x0C	SPH	--	--	--	--	--	(SP10)h	SP9	SP8	12
0x0B	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x0A	Reserved	--	--	--	--	--	--	--	--	
0x09	Reserved	--	--	--	--	--	--	--	--	
0x08	Reserved	--	--	--	--	--	--	--	--	
0x07	Reserved	--	--	--	--	--	--	--	--	
0x06	Reserved	--	--	--	--	--	--	--	--	
0x05	Reserved	--	--	--	--	--	--	--	--	
0x04	Reserved	--	--	--	--	--	--	--	--	
0x03	Reserved	--	--	--	--	--	--	--	--	
0x02	Reserved	--	--	--	--	--	--	--	--	
0x01	Reserved	--	--	--	--	--	--	--	--	
0x00	Reserved	--	--	--	--	--	--	--	--	
0x0F	ACSR	ACD	ACBG	ACD	ACI	ACIE	ACIC	ACIS1	ACIS0	248
0x0E	Reserved	--	--	--	--	--	--	--	--	
0x0D	SPDR	SPI Data Register								176
0x0C	SPSR	SPIF	WCOL	--	--	--	--	--	SP2X	175
0x0B	SPCR	SPE	SPE	DORF	MSTR	CPOL	CPHA	SPR1	SPR0	174
0x0A	GPICR2	General Purpose I/O Register 2								25
0x09	GPICR1	General Purpose I/O Register 1								25
0x08	Reserved	--	--	--	--	--	--	--	--	
0x07	OCRB	Timer/Counter Output Compare Register B								
0x06	OCRBA	Timer/Counter Output Compare Register A								
0x05	Reserved	--	--	--	--	--	--	--	--	
0x04	Reserved	--	--	--	--	--	--	--	--	
0x03	Reserved	--	--	--	--	--	--	--	--	
0x02	Reserved	--	--	--	--	--	--	--	--	
0x01	Reserved	--	--	--	--	--	--	--	--	
0x00	Reserved	--	--	--	--	--	--	--	--	
0x0F	EECR	--	--	EEM1	EEM0	EERIE	EEMPE	EEPE	EERE	21
0x0E	GPICR0	General Purpose I/O Register 0								25





## ANEXO 8

### Instrucciones del ATmega328P

#### ATmega48A/48PA/88A/88PA/168A/168PA/328/328P

### 5. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
<b>ARITHMETIC AND LOGIC INSTRUCTIONS</b>					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADW	Rd, K	Add Immediate to Word	$RdH:RdL \leftarrow RdH:RdL + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBW	Rd, K	Subtract Immediate from Word	$RdH:RdL \leftarrow RdH:RdL - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \wedge Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \wedge K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \text{one}'s\text{-}Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \text{two}'s\text{-}Rd$	Z,C,N,V,H	1
SBFR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \wedge (\text{one}'s\text{-}K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \wedge Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \wedge Rd$	Z,N,V	1
SEI	Rd	Set Register	$Rd \leftarrow \text{one}'s\text{-}Rd$	None	1
MUL	Rd, Rr	Multiply Unsigned	$Rr:Rd \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$Rr:Rd \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$Rr:Rd \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$Rr:Rd \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$Rr:Rd \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$Rr:Rd \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
<b>BRANCH INSTRUCTIONS</b>					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
LMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP <sup>(1)</sup>	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL <sup>(1)</sup>	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow \text{STACK}$	None	4
RETI		Interrupt Return	$PC \leftarrow \text{STACK}$	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	$\text{if } (Rd = Rr) \text{ then } PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
CP	Rd,Rr	Compare	$Rd - Rr$	Z,N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z,N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	$Rd - K$	Z,N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	$\text{if } (Rr[b]=0) \text{ then } PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	$\text{if } (Rr[b]=1) \text{ then } PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBSC	Rr, b	Skip if Bit in I/O Register Cleared	$\text{if } (Rr[b]=0) \text{ then } PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBSS	Rr, b	Skip if Bit in I/O Register is Set	$\text{if } (Rr[b]=1) \text{ then } PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	$\text{if } (SREG[s]=1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	$\text{if } (SREG[s]=0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BREQ	k	Branch if Equal	$\text{if } (Z = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRNE	k	Branch if Not Equal	$\text{if } (Z = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRCS	k	Branch if Carry Set	$\text{if } (C = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRCC	k	Branch if Carry Cleared	$\text{if } (C = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRSH	k	Branch if Same or Higher	$\text{if } (C = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRLO	k	Branch if Lower	$\text{if } (C = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRM	k	Branch if Minus	$\text{if } (N = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRPL	k	Branch if Plus	$\text{if } (N = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	$\text{if } (N \oplus V = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	$\text{if } (N \oplus V = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRHS	k	Branch if Half Carry Flag Set	$\text{if } (H = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	$\text{if } (H = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRIS	k	Branch if I Flag Set	$\text{if } (I = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRIC	k	Branch if I Flag Cleared	$\text{if } (I = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRVS	k	Branch if Overflow Flag is Set	$\text{if } (V = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	$\text{if } (V = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2

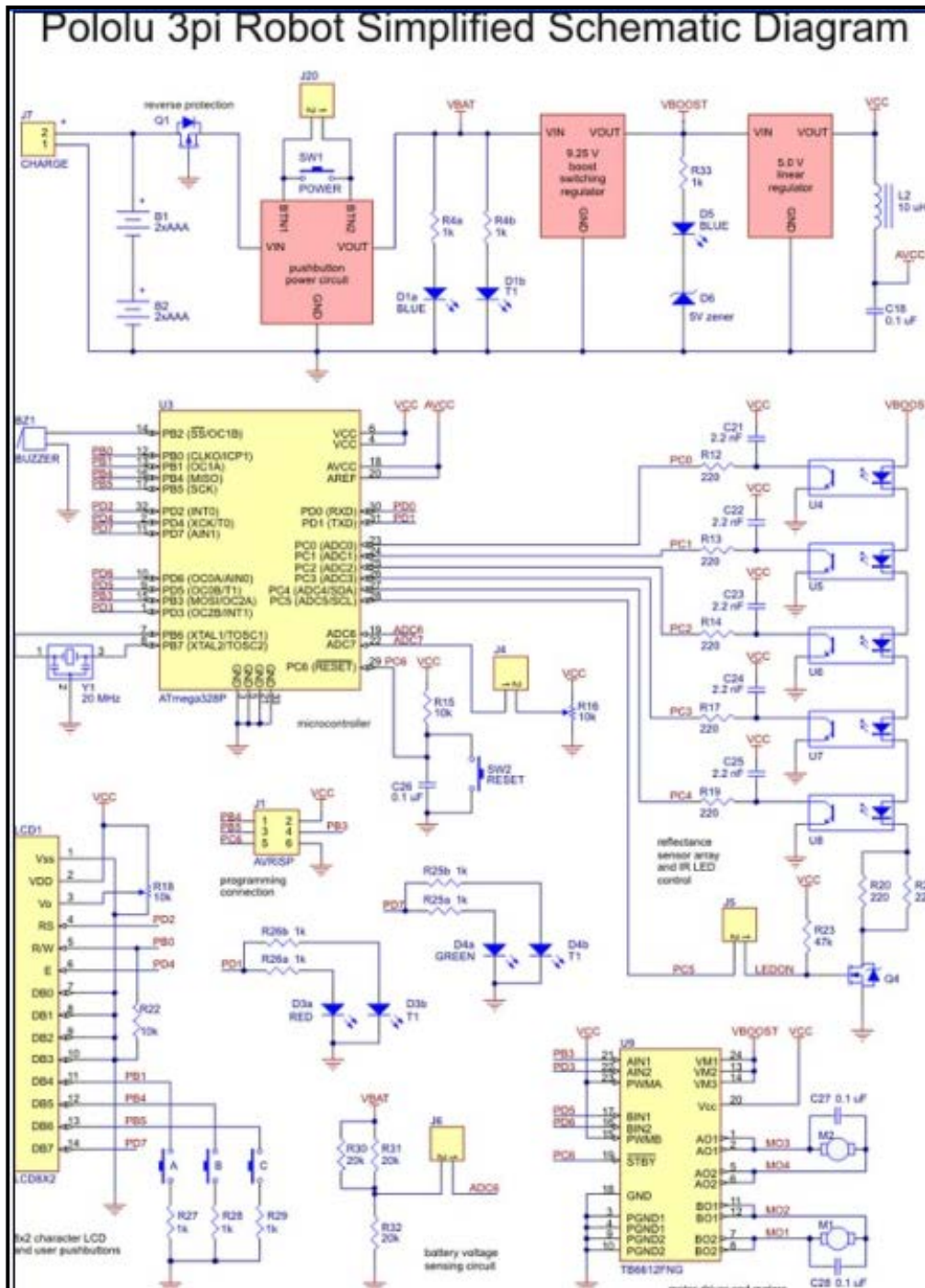
## ANEXO 9

### Instrucciones del ATmega328P

ATmega48A/48PA/88A/88PA/168A/168PA/328/328P						
Mnemonics	Operands	Description	Operation	Flags	#Clocks	
BRIE	k	Branch if Interrupt Enabled	$\text{if } (I = 1) \text{ then PC} \leftarrow \text{PC} + k + 1$	None	1/2	
BRID	k	Branch if Interrupt Disabled	$\text{if } (I = 0) \text{ then PC} \leftarrow \text{PC} + k + 1$	None	1/2	
<b>BIT AND BIT-TEST INSTRUCTIONS</b>						
SBI	P,D	Set Bit in I/O Register	$\text{I/O}(P,D) \leftarrow 1$	None	2	
CBI	P,D	Clear Bit in I/O Register	$\text{I/O}(P,D) \leftarrow 0$	None	2	
LSL	Rd	Logical Shift Left	$\text{Rd}(n+1) \leftarrow \text{Rd}(n), \text{Rd}(0) \leftarrow 0$	Z,C,N,V	1	
LSR	Rd	Logical Shift Right	$\text{Rd}(0) \leftarrow \text{Rd}(n+1), \text{Rd}(7) \leftarrow 0$	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	$\text{Rd}(0) \leftarrow \text{C}, \text{Rd}(n+1) \leftarrow \text{Rd}(n), \text{C} \leftarrow \text{Rd}(7)$	Z,C,N,V	1	
ROR	Rd	Rotate Right Through Carry	$\text{Rd}(7) \leftarrow \text{C}, \text{Rd}(n) \leftarrow \text{Rd}(n+1), \text{C} \leftarrow \text{Rd}(0)$	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	$\text{Rd}(0) \leftarrow \text{Rd}(n+1), n=0..6$	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	$\text{Rd}(3..0) \leftarrow \text{Rd}(7..4), \text{Rd}(7..4) \leftarrow \text{Rd}(3..0)$	None	1	
BSET	s	Flag Set	$\text{SREG}(s) \leftarrow 1$	$\text{SREG}(s)$	1	
BCLR	s	Flag Clear	$\text{SREG}(s) \leftarrow 0$	$\text{SREG}(s)$	1	
BST	Rt, b	Bit Store from Register to T	$T \leftarrow \text{Rt}(b)$	T	1	
BLD	Rd, b	Bit load from T to Register	$\text{Rd}(b) \leftarrow T$	None	1	
SEC		Set Carry	$C \leftarrow 1$	C	1	
CLC		Clear Carry	$C \leftarrow 0$	C	1	
SEN		Set Negative Flag	$N \leftarrow 1$	N	1	
CLN		Clear Negative Flag	$N \leftarrow 0$	N	1	
SEZ		Set Zero Flag	$Z \leftarrow 1$	Z	1	
CLZ		Clear Zero Flag	$Z \leftarrow 0$	Z	1	
SEI		Global Interrupt Enable	$I \leftarrow 1$	I	1	
CLI		Global Interrupt Disable	$I \leftarrow 0$	I	1	
SES		Set Signed Test Flag	$S \leftarrow 1$	S	1	
CLS		Clear Signed Test Flag	$S \leftarrow 0$	S	1	
SEV		Set Two's Complement Overflow	$V \leftarrow 1$	V	1	
CLV		Clear Two's Complement Overflow	$V \leftarrow 0$	V	1	
SET		Set T in SREG	$T \leftarrow 1$	T	1	
CLT		Clear T in SREG	$T \leftarrow 0$	T	1	
SEH		Set Half Carry Flag in SREG	$H \leftarrow 1$	H	1	
CLH		Clear Half Carry Flag in SREG	$H \leftarrow 0$	H	1	
<b>DATA TRANSFER INSTRUCTIONS</b>						
MOV	Rd, Rr	Move Between Registers	$\text{Rd} \leftarrow \text{Rr}$	None	1	
MOVW	Rd, Rr	Copy Register Word	$\text{RD}+1:\text{RD} \leftarrow \text{RR}+1:\text{RR}$	None	1	
LDI	Rd, K	Load Immediate	$\text{Rd} \leftarrow K$	None	1	
LDD	Rd, X	Load Indirect	$\text{Rd} \leftarrow [X]$	None	2	
LD	Rd, X+	Load Indirect and Post-Inc.	$\text{Rd} \leftarrow [X], X \leftarrow X + 1$	None	2	
LD	Rd, -X	Load Indirect and Pre-Dec.	$X \leftarrow X - 1, \text{Rd} \leftarrow [X]$	None	2	
LD	Rd, Y	Load Indirect	$\text{Rd} \leftarrow [Y]$	None	2	
LD	Rd, Y+	Load Indirect and Post-Inc.	$\text{Rd} \leftarrow [Y], Y \leftarrow Y + 1$	None	2	
LD	Rd, -Y	Load Indirect and Pre-Dec.	$Y \leftarrow Y - 1, \text{Rd} \leftarrow [Y]$	None	2	
LDD	Rd, Y+q	Load Indirect with Displacement	$\text{Rd} \leftarrow [Y + q]$	None	2	
LD	Rd, Z	Load Indirect	$\text{Rd} \leftarrow [Z]$	None	2	
LD	Rd, Z+	Load Indirect and Post-Inc.	$\text{Rd} \leftarrow [Z], Z \leftarrow Z + 1$	None	2	
LD	Rd, -Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1, \text{Rd} \leftarrow [Z]$	None	2	
LDD	Rd, Z+q	Load Indirect with Displacement	$\text{Rd} \leftarrow [Z + q]$	None	2	
LDS	Rd, k	Load Direct from SRAM	$\text{Rd} \leftarrow [k]$	None	2	
ST	X, Rr	Store Indirect	$[X] \leftarrow \text{Rr}$	None	2	
ST	X+, Rr	Store Indirect and Post-Inc.	$[X] \leftarrow \text{Rr}, X \leftarrow X + 1$	None	2	
ST	-X, Rr	Store Indirect and Pre-Dec.	$X \leftarrow X - 1, [X] \leftarrow \text{Rr}$	None	2	
ST	Y, Rr	Store Indirect	$[Y] \leftarrow \text{Rr}$	None	2	
ST	Y+, Rr	Store Indirect and Post-Inc.	$[Y] \leftarrow \text{Rr}, Y \leftarrow Y + 1$	None	2	
ST	-Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1, [Y] \leftarrow \text{Rr}$	None	2	
STD	Y+q, Rr	Store Indirect with Displacement	$[Y + q] \leftarrow \text{Rr}$	None	2	
ST	Z, Rr	Store Indirect	$[Z] \leftarrow \text{Rr}$	None	2	
ST	Z+, Rr	Store Indirect and Post-Inc.	$[Z] \leftarrow \text{Rr}, Z \leftarrow Z + 1$	None	2	
ST	-Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1, [Z] \leftarrow \text{Rr}$	None	2	
STD	Z+q, Rr	Store Indirect with Displacement	$[Z + q] \leftarrow \text{Rr}$	None	2	
STS	k, Rr	Store Direct to SRAM	$[k] \leftarrow \text{Rr}$	None	2	
LPM		Load Program Memory	$\text{R0} \leftarrow [Z]$	None	3	
LPM	Rd, Z	Load Program Memory	$\text{Rd} \leftarrow [Z]$	None	3	
LPM	Rd, Z+	Load Program Memory and Post-Inc.	$\text{Rd} \leftarrow [Z], Z \leftarrow Z + 1$	None	3	
SPM		Store Program Memory	$[Z] \leftarrow \text{Rr}, \text{R0}$	None	-	
IN	Rd, P	In Port	$\text{Rd} \leftarrow P$	None	1	
OUT	P, Rr	Out Port	$P \leftarrow \text{Rr}$	None	1	
PUSH	Rr	Push Register on Stack	$\text{STACK} \leftarrow \text{Rr}$	None	2	


# ANEXO 10

Esquemático del Pololu 3π



## ANEXO 11

### Comandos AT acorde con GSM

SIM340DZ AT Commands Set	
	
<b>3 AT Commands According to GSM07.07</b>	
<b>3.1 Overview of AT Command According to GSM07.07</b>	
Command	Description
AT+CACM	ACCUMULATED CALL METER(ACM) RESET OR QUERY
AT+CAMM	ACCUMULATED CALL METER MAXIMUM(ACM MAX) SET OR QUERY
AT+CAOC	ADVICE OF CHARGE
AT+CBST	SELECT BEARER SERVICE TYPE
AT+CCFC	CALL FORWARDING NUMBER AND CONDITIONS CONTROL
AT+CCUG	CLOSED USER GROUP CONTROL
AT+CCWA	CALL WAITING CONTROL
AT+CEER	EXTENDED ERROR REPORT
AT+CGMI	REQUEST MANUFACTURER IDENTIFICATION
AT+CGMM	REQUEST MODEL IDENTIFICATION
AT+CGMR	REQUEST TA REVISION IDENTIFICATION OF SOFTWARE RELEASE
AT+CGSN	REQUEST PRODUCT SERIAL NUMBER IDENTIFICATION (IDENTICAL WITH +GSN)
AT+CSCS	SELECT TE CHARACTER SET
AT+CSTA	SELECT TYPE OF ADDRESS
AT+CHLD	CALL HOLD AND MULTIPARTY
AT+CDMI	REQUEST INTERNATIONAL MOBILE SUBSCRIBER IDENTITY
AT+CKPD	KEYPAD CONTROL
AT+CLCC	LIST CURRENT CALLS OF ME
AT+CLCK	FACILITY LOCK
AT+CLIP	CALLING LINE IDENTIFICATION PRESENTATION
AT+CLIR	CALLING LINE IDENTIFICATION RESTRICTION
AT+CMEE	REPORT MOBILE EQUIPMENT ERROR
AT+COLP	CONNECTED LINE IDENTIFICATION PRESENTATION
AT+COPS	OPERATOR SELECTION
AT+CPAS	MOBILE EQUIPMENT ACTIVITY STATUS
AT+CPBF	FIND PHONEBOOK ENTRIES
AT+CPBR	READ CURRENT PHONEBOOK ENTRIES
AT+CPBS	SELECT PHONEBOOK MEMORY STORAGE
AT+CPBW	WRITE PHONEBOOK ENTRY
AT+CPIN	ENTER PIN
AT+CPWD	CHANGE PASSWORD
AT+CR	SERVICE REPORTING CONTROL

SIM340DZ\_ATC\_V1.02 29.08.2008

37

## ANEXO 12

### Comandos AT acorde con GSM

SIM340DZ AT Commands Set	
AT+CRIC	SET CELLULAR RESULT CODES FOR INCOMING CALL INDICATION
AT+CREG	NETWORK REGISTRATION
AT+CRLP	SELECT RADIO LINK PROTOCOL PARAMETER
AT+CRSM	RESTRICTED SIM ACCESS
AT+CSQ	SIGNAL QUALITY REPORT
AT+FCLASS	FAX: SELECT, READ OR TEST SERVICE CLASS
AT+FMI	FAX: REPORT MANUFACTURED ID
AT+FMM	FAX: REPORT MODEL ID
AT+FMR	FAX: REPORT REVISION ID
AT+VTD	tone DURATION
AT+VTS	DTMF AND TONE GENERATION
AT+CMUX	MULTIPLEXER CONTROL
AT+CNUM	SUBSCRIBER NUMBER
AT+CPOL	PREFERRED OPERATOR LIST
AT+OOPN	READ OPERATOR NAMES
AT+CFUN	SET PHONE FUNCTIONALITY
AT+OCLK	CLOCK
AT+CSIM	GENERIC SIM ACCESS
AT+CALM	ALERT SOUND MODE
AT+CRSL	RINGER SOUND LEVEL
AT+CLVL	LOUD SPEAKER VOLUME LEVEL
AT+CMUT	MUTE CONTROL
AT+CPUC	PRICE PER UNIT CURRENCY TABLE
AT+OCWE	CALL METER MAXIMUM EVENT
AT+CBC	BATTERY CHARGE
AT+CUSD	UNSTRUCTURED SUPPLEMENTARY SERVICE DATA
AT+CSSN	SUPPLEMENTARY SERVICES NOTIFICATION

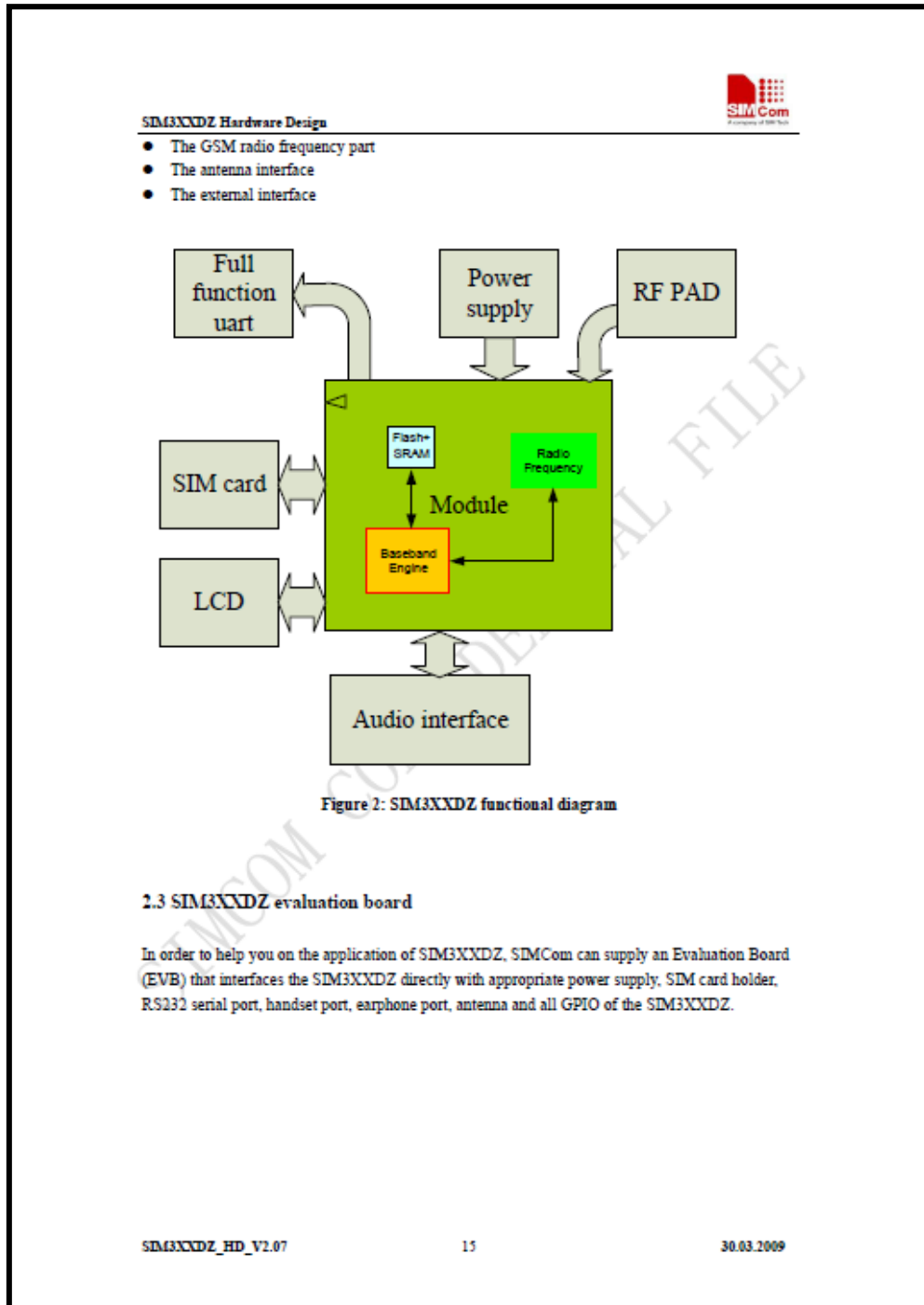
3.2 Detailed Descriptions of AT Command According to GSM07.07	
3.2.1 AT+CACM Accumulated Call Meter (ACM) Reset Or Query	
AT+CACM Accumulated Call Meter (ACM) Reset Or Query	
Test Command AT+CACM=?	Response OK Parameter
Read Command AT+CACM?	Response TA returns the current value of ACM. +CACM: <acm> OK

SIM340DZ\_ATC\_V1.02
38
29.08.2008




## ANEXO 13

### Diagrama de bloques del modulo SIM304DZ



## ANEXO 14

### Descripción de Pines

  
SIMCom  
University of Granada

---

### 3 Application interface

All hardware interfaces are described in detail in following chapters:

- Power supply and charging control (see Chapters 3.3 and 3.5)
- Provide serial interface and Debug interface (see chapter 3.9)
- Two analog audio interfaces (see chapter 3.10)
- SIM interface (see chapter 3.11)

#### 3.1 SIM3XXDZ Pin description


**Table 5: Pin description**

Power Supply				
PIN NAME	I/O	DESCRIPTION	DC CHARACTERISTICS	COMMENT
VBAT		2 VBAT pins are dedicated to connect the supply voltage. The power supply of SIM3XXDZ has to be a single voltage source of VBAT= 3.4V...4.5V. It must be able to provide sufficient current in a transmit burst which typically rises to 2A mostly, these 2 pins are voltage input, however ,when use the charge circuit to charge the battery ,these pins become the current output, select one of these pins as the charge current output Pin	Vmax= 4.5V Vmin=3.4V Vnorm=4.0V	
VRTC	I/O	Current input for RTC when the battery is not supplied for the system. Current output for backup	Vmax=2.0V Vmin=1.2V Vnorm=1.8V Iout(max)= 20uA	Do not keep Pin open, it should be connected to a battery or a

SIM3XXDZ\_HD\_V2.07 17 30.03.2009

## ANEXO 15

### Descripción de Pines




**SIM3XXDZ Hardware Design**

		battery when the main battery is present and the backup battery is in low voltage state.	$I_{in}=5\ \mu A$	capacitor.
VCHG	I	Voltage input for the charge circuit, as the signal for detecting the charger connecting	$V_{max}=5.25V$ $V_{min}=1.1$ VBAT $V_{norm}=5.1V$ $I_{min}=650mA$	If unused keep Pin open
GND		Digital ground		
<b>Power on or power off</b>				
PIN NAME	I/O	DESCRIPTION	DC CHARACTERISTICS	COMMENT
PWRKEY	I	Voltage input for power on key. Press the key , the PWRKEY get a low level voltage for user to power on or power off the system, the user should keep pressing the key for a moment when power on or power off the system. Because the system need margin time assert the software.	$V_{Lmax}=0.2*V_{BAT}$ $V_{Hmin}=0.6*V_{BAT}$ $V_{Lmax}=V_{BAT}$	Pull up to VBAT inside.
<b>Audio interfaces</b>				
PIN NAME	I/O	DESCRIPTION	DC CHARACTERISTICS	COMMENT
MIC1P MIC1N	I	Positive and negative voiceband input	Audio DC Characteristics refer to chapter 3.10	If unused keep Pin open
MIC2P MIC2N	I	Auxiliary positive and negative voiceband input		If unused keep Pin open
SPK1P SPK1N	O	Positive and negative voiceband output		If unused keep Pin open
SPK2P SPK2N	O	Auxiliary positive and negative voiceband output		If unused keep Pin open
AGND		Analog ground		Separate ground connection for external audio

SIM3XXDZ\_HD\_V2.07 18 30.03.2009

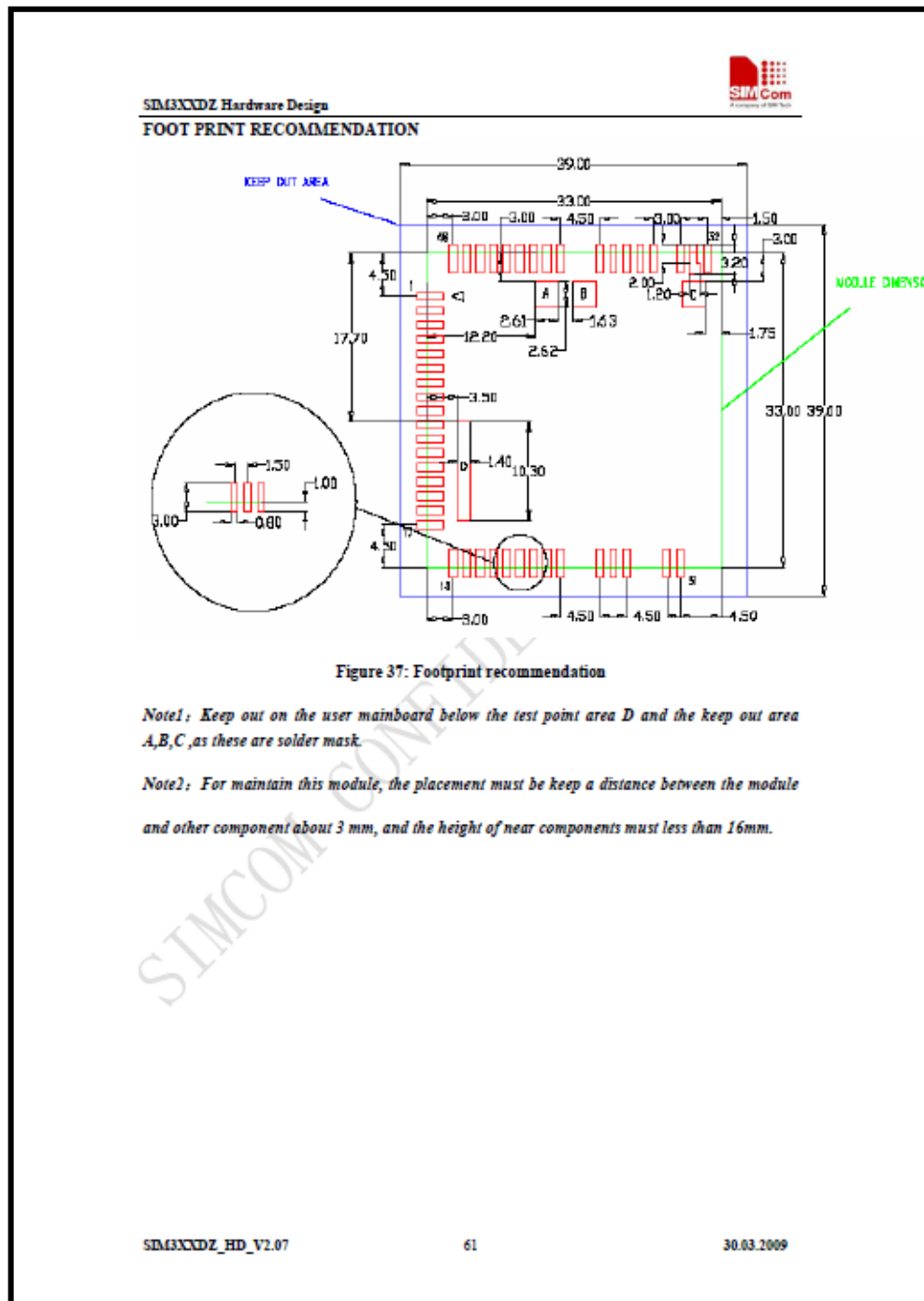
## ANEXO 16

### Descripción de Pines

SIM3XXDZ Hardware Design				
				circuits. If unused keep Pin open
GERNERAL PURPOSE input/output				
PIN NAME	I/O	DESCRIPTION	DC CHARACTERISTICS	COMMENT
STATUS	O	Indicate work status	VILmin=0V VILmax=0.3*2.93V	If unused keep pins open
GPO1	O	Normal Output Port	VIHmin=0.7*2.93V	If unused keep pins open
DISP_DATA	I/O /4 mA	Display interface	VOLmin=GND VOLmax=0.2V VOHmin=2.93V-0.2 VOHmax=2.93V	If unused keep pins open
DISP_CLK	O/4 mA			
DISP_CS	O/4 mA			
DISP_D/C	O/4 mA			
DISP_RST	O/4 mA			
KBR0	I/4 mA			
Serial interface				
PIN NAME	I/O	DESCRIPTION	DC CHARACTERISTICS	COMMENT
RXD	I/8 mA	Receive data	VILmin=0V VILmax=0.3*2.93V	If use only TXD, RXD GND three pins to communicate, RTS Pin connect to GND directly. DTR Pin is pulled up inside. If unused keep pins open
DTR	I/8 mA	Data terminal Ready	93V VIHmin=0.7*2.93V	
TXD	O/8 mA	Transmit data	93V VIHmax=2.93V+0.3	
RTS	I/8 mA	Request to send	VOLmin=GND	
CTS	O/8 mA	Clear to send	VOLmax=0.2V	
RI	O/8 mA	Ring indicator	VOHmin=2.93V-0.2 VOHmax=2.93V	


## ANEXO 17

### Vista superior del modulo SIM



## ANEXO 18

### Asignación de Pines del modulo SIM

  
Company of SIM Tech

**SIM3XXDZ Hardware Design**

---

#### 6.2 PIN assignment of SIM3XXDZ

**Table 29: PIN assignment**

Pin NUM	NAME	Pin NUM	NAME
1	DBG_RXD	36	GND
2	DBG_TXD	37	GND
3	RXD	38	VBAT
4	TXD	39	VBAT
5	STATUS	40	GPO1
6	SDM_DATA	41	NETLIGHT
7	SDM_CLK	42	DCD
8	SDM_RST	43	DTR
9	SDM_VDD	44	RTS
10	KBR0	45	CTS
11	RI	46	DISP_CS
12	PWRKEY	47	NC
13	DISP_CLK	48	GND
14	DISP_DATA		
15	VRTC		
16	DISP_D/C		
17	GND		
18	MIC2P		
19	MIC2N		
20	MIC1N		
21	MIC1P		
22	AGND		
23	SPK1P		
24	SPK1N		
25	SPK2N		
26	SPK2P		
27	TEMP_BAT		
28	VCHG		
29	ADC0		
30	GND		
31	GND		
32	GND		
33	ANTENNA		
34	GND		

SIM3XXDZ\_HD\_V2.07 62 30.03.2009

## ANEXO 19

### Código Principal del Proyecto

```
/*
*****
***** CODIGO PRINCIPAL *****
*****
*/
#include <avr/io.h>
#include "timer640.h"
#include "uart4.h"
#include "rprintf.h"
#include "dronecell.h"
#include "buffer.h"
#include "sor_utils.h"
#include <Pololu/3π.h>

void FlushReceiveBuffer(void);
unsigned char USART_Receive( void );

int main(void)
{

// i, j: variables
// band: bandera para indicar la orientación de los giros
int i,j,band;
int temp, temp1, temp2; // temporales para el calculo del tiempo de los motores
int ErrorCode=0; //variable para validar el inicio del DroneCell
char button;

    configure_ports();

    do
    {
        clear();
        lcd_init_printf();
        print("Press B");
        lcd_goto_xy(0, 1);
        print("to Start");
        button = wait_for_button_press(ALL_BUTTONS);

        if(button & BUTTON_B)
        {
            temp=1;

        } else {temp=0;}

    }while(temp==0); // este do while sirve para que al momento de presionar la botonera
B se inicie el programa
    uart0Init();

    uartSetBaudRate(0, 115200);
    //uartSetBaudRate(0, 9600); //para simulación en proteus
```

```

rprintfInit(uart0SendByte);
init_timer0(TIMER_CLK_64);

//delay_ms_narobo(2000);
delay_ms_narobo(2000);
clear();
lcd_init_printf();
print("Iniciando");
lcd_goto_xy(0, 1);
print("DroneCell");

DroneCell_Init( uart0SendByte , uart0GetByte , FlushReceiveBuffer );
delay_ms_narobo(2000);
delay_ms_narobo(2000);

clear();
lcd_init_printf();
print("DroneCell");
lcd_goto_xy(0, 1);
print("Iniciado");

ErrorCode = DroneCell_PowerOn_Pololu();
    if ( ErrorCode == OK) {

        clear();
        lcd_init_printf();
        print("DroneCell");
        lcd_goto_xy(0, 1);
        print("Listo");
    }
    else
    {
        clear();
        lcd_init_printf();
        print("DroneCell");
        lcd_goto_xy(0, 1);
        print("Error");
    }

    rprintf("AT+CMGDA=");
    uartSendByte(0,34);
    rprintf("DEL ALL");
    uartSendByte(0,34);
    rprintf("\n");

    delay_ms_narobo(2000);
    ErrorCode = DroneCell_Receive_TextInit(); //Función para recibir el mensaje de
texto

    if ( ErrorCode != OK)
    {
        clear();

```



```

        lcd_init_printf();
        print("Error");
        lcd_goto_xy(0, 1);
        print("de texto");
    }
    else
    {
        clear();
        lcd_init_printf();
        print("Espero");
        lcd_goto_xy(0, 1);
        print("el SMS");
    }
}

/*****
/***** INICIO DEL WHILE *****/
/*****/

while(1){
    delay_ms_narobo(2000);
    uartFlushReceiveBuffer(0);
    delay_ms_narobo(3000);
    while( (uart0GetByte() != -1) ) //verifica si llego el mensaje , -1 indica que no llega
    todavia el mensaje
        clear();
        lcd_init_printf();
        print("SMS");
        lcd_goto_xy(0, 1);
        print("Recibido");

        delay_ms_narobo(2000);
        for(i=0;i<47;i++) //este for sirve para empezar a leer el mensaje desde la instruccion ya
        que en el buffer se almacena 46 caracteres que no son de utilidad para nuestro objetivo, solo
        sirven de respuesta
            {
                temp=uart0GetByte();
                uartSendByte(0,temp);
            }
        temp='U'; // inicializo temp en U para poder ingresar al while
        band=0; // inicializo band para empezar con una instruccion que diga hacia adelante

/*****/
/*****/
// Este while valida las instrucciones para el movimiento del Pololu 3π.
// U o u: Significa que el Pololu 3π se dirija hacia adelante
// D o d: Significa que el Pololu 3π se dirija hacia atrás
// L o l: Significa que el Pololu 3π se dirija hacia la izquierda
// R o r: Significa que el Pololu 3π se dirija hacia la derecha
// También el mensaje tiene un formato definido el cual se indica con un ejemplo
// U50L20D05R90, eso significa que el Pololu 3π se va a trasladar 50 segundos adelante, 20
segundos hacia la izquierda, luego

// 5 segundos hacia la atrás y finalmente 90 segundos a la derecha. Cabe señalar que el
mensaje puede tener más instrucciones

```

// de movimiento o sea después de R90 se puede seguir con mas movimientos y también el mensaje esta validado para que se pueda recibir  
// la instrucción con letras minúsculas o sea el mensaje de ejemplo también puede escribirse de la siguiente manera a50I20d05r90.  
// El numero después de la letra tienen que ser de dos dígitos ejemplo para enviar al Pololu que se mueva 5 segundos  
// debe enviarse 05 y no solo 5. La velocidad del Pololu ya esta predefinida por el autor.

```
while((temp>=48&&temp<=57)||temp=='U'||temp=='r'||temp=='D'||temp=='d'||temp
=='L'||temp=='l'||temp=='R'||temp=='r')
{
    temp=uart0GetByte();
    uartSendByte(0,temp);
```

//\*\*\*\*\* U==UP: MOVER POLOLU HACIA ADELANTE \*\*\*\*\*

```
if(temp=='U' || temp=='u')
{
    temp=uart0GetByte(); // obtiene un byte almacenado en el buffer
    if((temp>=48&&temp<=57))//
    {
        temp1=10*(temp-48); // guardo el primer digito,
se resta 48 debido a que el 0 en decimal es 48
        temp=uart0GetByte();
        if((temp>=48&&temp<=57))
        {
            clear();
            lcd_init_printf();
            print("Pololu 3π");
            lcd_goto_xy(0, 1);
            print("to UP");

            temp2=(temp1)+(temp-48);// en temp2 se
almacena el número total del tiempo para empezar a mover al Pololu
            set_motors(38, 38);// arranco los motores
            for(j=0;j<1000;j++){
                delay_ms_narobo(temp2);}
        }
    }
    band=0;
} //*****FIN DE MOVIMIENTO HACIA ADELANTE
```

//\*\*\*\*\* D==DOWN: MOVER POLOLU HACIA ATRAS \*\*\*\*\*

```
if(temp=='D' || temp=='d')
{
    temp=uart0GetByte();
    if((temp>=48&&temp<=57))
    {
        temp1=10*(temp-48);

        temp=uart0GetByte();
        if((temp>=48&&temp<=57))
        {
            clear();
```

```

        lcd_init_printf();
        print("Pololu 3π");
        lcd_goto_xy(0, 1);
        print("to DOWN");
        temp2=(temp1)+(temp-48);
        set_motors(-38, -38);
        for(j=0;j<1000;j++){
            delay_ms_narobo(temp2);}
    }
}
band=1;
}/**FIN DE MOVIMIENTO HACIA ATRAS

/*******
L==LEFT:  GIRO  DEL  POLOLU  HACIA  LA  IZQUIERDA
*****

if(temp=='L' || temp=='l')
{
    if(band==0){
        set_motors(0,38);
        for(j=0;j<1000;j++){
            delay_ms_narobo(1);}
        temp=uart0GetByte();
        if((temp>=48&&temp<=57))
        {
            temp1=10*(temp-48);
            temp=uart0GetByte();
            if((temp>=48&&temp<=57))
            {
                clear();
                lcd_init_printf();
                print("Pololu 3π");
                lcd_goto_xy(0, 1);
                print("to LEFT");
                temp2=(temp1)+(temp-48);
                set_motors(38, 38);
                for(j=0;j<1000;j++){
                    delay_ms_narobo(temp2);}
            }
        }
    }/**FIN DE MOVIMIENTO HACIA LA IZQUIERDA con band==0
else
{
    set_motors(-38,0);
    for(j=0;j<1000;j++){
        delay_ms_narobo(1);}
    temp=uart0GetByte();
    if((temp>=48&&temp<=57))

{
    temp1=10*(temp-48);

```



```

        if((temp>=48&&temp<=57))
        {
            temp1=10*(temp-48);
            temp=uart0GetByte();
            if((temp>=48&&temp<=57))
            {

                clear();
                lcd_init_printf();
                print("Pololu 3π");
                lcd_goto_xy(0, 1);
                print("to RIGHT");

                temp2=(temp1)+(temp-48);
                set_motors(-38, -38);
                for(j=0;j<1000;j++){
                    delay_ms_narobo(temp2);}
            }
        }
    }//*****FIN DE MOVIMIENTO HACIA LA DERECHA con band==1
} //*****FIN DE MOVIMIENTO HACIA LA DERECHA (temp=='D' | temp=='d')

    temp=uart0GetByte();
    clear();
    lcd_init_printf();
    print("Espero");
    lcd_goto_xy(0, 1);
    print("Nue. SMS");
    set_motors(0, 0); //PARO LOS MOTORES INDICANDO QUE SE
TERMINO LA INSTRUCCION DEL MENSAJE
} //**** FIN DEL WHILE
rprintf("FIN\n");
} //***** FIN DE MAIN *****

void FlushReceiveBuffer(void) { // necessary to put here so that FlushBuffer pointer works
    uartFlushReceiveBuffer(0);
}
unsigned char USART_Receive( void )
{
    /* Wait for data to be received */
    while ( !(UCSROA & (1<<RXCO)) )
    ;
    /* Get and return received data from buffer */
    return UDR0; }

```

# BIBLIOGRAFIA

1. - LabCenter Electronics – PCB Design.

[http://www.labcenter.com/products/pcb\\_overview.cfm](http://www.labcenter.com/products/pcb_overview.cfm)

[http://es.wikipedia.org/wiki/Proteus\\_%28electr%C3%B3nica%29](http://es.wikipedia.org/wiki/Proteus_%28electr%C3%B3nica%29)

Fecha de Consulta: 17/01/2011

2.- Pololu 3π – Características.

<http://www.Pololu.com/catalog/product/975>

Fecha de Consulta: 18/01/2011

3.- Pololu 3π – Guía de Usuario.

<http://www.Pololu.com/file/OJ137/Pololu3πRobotGuiaUsuario.pdf>

Fecha de Consulta: 18/01/2011

4.- Narobo DroneCell – Características.

<http://narobo.com/products/DroneCell/DroneCell.html>

Fecha de Consulta: 18/01/2011

5.- Narobo DroneCell – videos.

- Send a Text Message

[www.youtube.com%2Fwatch%3Fv%3DL8eSg5MRlzk](http://www.youtube.com%2Fwatch%3Fv%3DL8eSg5MRlzk)

- Phone Call

[www.youtube.com%2Fwatch%3Fv%3D7is4PnLacCA](http://www.youtube.com%2Fwatch%3Fv%3D7is4PnLacCA)

Fecha de Consulta: 22/01/2011

6.- Narobo DroneCell – Guía de Comandos.

[http://narobo.com/products/DroneCell/Datasheets/SIM340DZ\\_ATC\\_V1.02.pdf](http://narobo.com/products/DroneCell/Datasheets/SIM340DZ_ATC_V1.02.pdf)

Fecha de Consulta: 20/01/2011

7.- Narobo DroneCell – Guía del Hardware.

[http://narobo.com/products/DroneCell/Datasheets/SIM340\\_HD.pdf](http://narobo.com/products/DroneCell/Datasheets/SIM340_HD.pdf)

Fecha de Consulta: 20/01/2011

8.- Narobo DroneCell – Guía de Mensajes de texto.

[http://narobo.com/products/DroneCell/Datasheets/SMS\\_Guide.pdf](http://narobo.com/products/DroneCell/Datasheets/SMS_Guide.pdf)

Fecha de Consulta: 19/01/2011

9.- AVR Studio.

[http://www.atmel.com/dyn/products/tools\\_card.asp?tool\\_id=2725](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725)

Fecha de Consulta: 15/01/2011

10.- AVR Studio – Guía de Usuario.

<http://courses.cit.cornell.edu/ee476/AtmelStuff/doc1019.pdf>

Fecha de Consulta: 15/01/2011

11.- ATmega328P

[http://www.atmel.com/dyn/resources/prod\\_documents/8271S.pdf](http://www.atmel.com/dyn/resources/prod_documents/8271S.pdf)

Fecha de Consulta: 15/01/2011

12.- SMS Tutorial: Introduction AT Commands.

<http://www.developershome.com/sms/atCommandsIntro.asp>

Fecha de Consulta: 23/01/2011