People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In **Electrical and Electronic Engineering**

Option: **Computer**

Title:

Design and Realization of Mobile Robot

Presented by:

- **BOUCHETIBAT AHCENE**
- **MEKKAKIA MAAZA YAHIA ABDESLEM**

Supervisor:

Mr. Zakarya GUETTATFI

Registration Number:……../2015

Dedication

IN THE NAME OF ALLAH THE MOST BENEFICIANT THE MERCIFULL

We want to dedicate our Project to

The Persons who are cause of our success,

These are our Loving and Caring Parents,

Who are always sources of success for us,

Who are always ready to push us for the better,

May God make us able to give the smallest part

Of what they gave us from our childhood to infinity.

Acknowledgement

We would like to express our great gratitude to our supervisor Mr. Zakarya GUETTATFI for his precious presence with us during the implementation of this work and for the great continuous guidance and supervision he offered through all steps.

It was a great honor for us to work with you Sir

A special thanks to our co-supervisor Dr. Abdelhakim KHOUAS for his suggestions and introducing us to Mr.Zakarya.

We also offer our regards and blessings to all of those who supported us in any aspect during the completion of the project.

Abstract

The main purpose of this work is to design and realize a mobile robot tricycle-platform; this robot is controlled remotely through a network by a graphical user interface. The robot sends its position to the graphical user interface for localization purposes. The robot motion is based on differential drive concept.

A microcontroller board called Arduino Mega 2560 is used as the brain of the tricycle-platform; this tricycle platform is designed using AutoCAD. Arduino Mega controls the operation of two motors, GPS, and a Wi-Fi Module. The two DC motors are used to move the robot. The GPS is used to locate and specify the mobile robot coordinates, these coordinates are used by the graphical user interface to show the current location of the robot on a Google Map. The Wi-Fi module is used to add networking capabilities to the mobile robot; a network is set up based on server/client architecture to exchange data and commands between the graphical user interface and the mobile robot. The graphical user interface is developed using Visual C# and is deployed on our personal computer.

As a conclusion, this work shows that differential drive concept is suitable for robots that are based on tricycle-platforms. The localization feature added by the GPS to the robot makes it reachable everywhere. Networking aspect is a good way for remotely controlling robots. And with all the powerful aspects offered by Visual Studio to develop a graphical user interface making it a good first choice to use.

Table of Contents

List of Figures

List OF Abbreviations

API Application Programming Interface

AVR Alf Vegard RISC

CCS Cascading Style Sheet

DC Direct Current

DSL Digital Subscriber Line

EEPROM Electrically Erasable Programmable Read-Only Memory

GPS Global Positioning System

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ICSP In-Circuit Serial Programming

IDE Interface Development Environment

IEEE Institute of Electrical and Electronics Engineers

JTAG Joint Test Action Group

LAN Local Area Network

NMEA National Marine Electronics Association

PROM Programmable Read Only Memory

PWM Pulse Width Modulation

RAM Random Access Memory

RISC Reduced Instruction Set Computing

ROM Read Only Memory

RTCM Radio Technical Commission for Maritime Services

SPI Serial Peripheral Interface

SSID Service Set Identifier

TCP/IP Transmission Control Protocol (TCP)/ Internet Protocol (IP)

TTL Transistor-Transistor Logic

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

UTC Universal Time Coordinates

WPF Windows Presentation Foundation

CHAPTER 1

Introduction

Here we introduce some generalities about robotics and we will talk about mobile robots and GPS localization and mapping.

1.1 OVERVIEW

Robotics is a field that is becoming widespread; it can cover different disciplines and even wide array of topics within one discipline. Because of this, designing a robot leads to many challenges to overcome and new things to learn. Due to the recent technological growth, this field has seen a large beneficiary of these advancements especially in Mobile robotics.

Today Mobile robotics finds application in many areas like automatic cleaning, agriculture, support to medical services, hazard environments, space exploration, military, intelligent transportation, social robotics, and entertainment.

From this analysis, we can understand that the need for practical integration tools to implement valuable scientific contributions is felt frequently. For that purpose, several different mobile robotic platforms have emerged with the ability to support research work focusing on applications like search and rescue, security applications or human interaction. Therefore, various mechanisms are used to control mobile robots.

Telerobotics is the area of robotics concerned with the control of semi-autonomous robots from a distance, chiefly using Wireless network (like Wi-Fi, Bluetooth, the Deep Space Network, and similar) or tethered connections.

One major subfield of telerobotics is **teleoperation.** It indicates operation of a machine at a distance. It is similar in meaning to the phrase "remote control" but is usually encountered in research, academic and technical environments. It is most commonly associated with robotics and mobile robots, but can be applied to a whole range of circumstances in which a device or machine is operated by a person from a distance.

Integrating mobile robots with GPS receivers is now widely used for outdoors navigation and has many applications such as positioning, locating, navigating, mapping, surveying and time determination. For outdoors navigation, GPS provides three dimensions position and velocity information. Under favorable conditions, GPS can provide continuous and relatively accurate navigation information over a long period of time. Much work has been carried out on the use of GPS-based navigation systems in mobile robots.

As we have seen, it is a fundamental requirement that a mobile robot should be able to know its position within its operating environment before any assigned task can be accomplished, for example; in space exploration, underwater navigation, warehouse and office deliveries. Furthermore, a priori knowledge of the environment may not always be available, such as in cases

of search and rescue operations, otherwise a cost has to be invested in structuring the environment. Therefore, it is important that the robot is able to build a map of its operating area.

1.2 Motivation

All these years passed studying electronics made us able to gain more knowledge about this field. Developing a mobile robotic platform with various sensors and capabilities is a challenge that we wanted to accomplish. With The structure of the robot and the nature of Arduino board, it also leads to being able to mount additional sensors for whatever might be needed in future applications.

1.3 Project Objectives

The purpose of the project is to design a mobile robot capable of performing several types of operations:

- Real time control of the robot motion remotely using a Graphical User Interface.
- Robot receives GPS information and sends them to a computer through network.
- Graphical User Interface receives, processes, and displays GPS data.
- Graphical User Interface shows the localization of the mobile robot using Google Maps API.
- Set up a network to exchange data, and create a server/client structure in which the robot is the server, and the computer is the client.

1.4 Organization of The Report

This Report is divided in four parts. The first chapter is an introduction to mobile robotic, telerobotics systems, the integration of GPS receivers and mapping. The second chapter describes the Theoretical Background to identify basic operations of all used components with their internal structure. The third chapter describes the Hardware Design, where we expose the most important work and we show how to interface all components together. Finally, the last chapter is the Software Design, where we demonstrate the different flowcharts of the programs.

CHAPTER II

Theoretical Background

This chapter introduces the theory and technical details of the different components used in the hardware design system, the theory side explains how different parts work and function. This chapter also introduces the mechanical description of the platform (Design).

2.1 Microcontrollers

A microcontroller (µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Some microcontrollers are designed to connect to a computer for programming for specialized purposes. ATmega2560 is an example of one of these microcontrollers. Microcontrollers make it easier to build electronic devices because the control of their functions is via code, they can control and interpret forms of both input and output, and are designed to fit specified criteria depending on the applications.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. [1]

2.1.1 Microcontrollers History

Since the inception of microprocessors, microcontrollers have developed and appeared in market. Intel 4004 was the first 4-bit processor which appeared in 1971. In 1974, Texas instrument introduced the first microcontroller TMS 1000 which has on-chip RAM, ROM, and I/Os. Most microcontrollers at that time had two variants: one had an erasable EEPROM program memory, which was significantly more expensive than the PROM variant which was only programmable once. However all required external chips to implement a working system, raising total system cost, and making it impossible to economically computerize appliances.

As Moore observed that the complexity and integration density of Microprocessors were augmenting dramatically with time, which contributes the development of powerful microcontrollers. Nowadays, we have super-universal microcontrollers with high performance and low cost. [2]

2.1.2 ATmega2560

ATmega2560 is a low-power Atmel 8-bit AVR RISC-based microcontroller combines 256KB ISP flash memory, 8KB SRAM, 4KB EEPROM, 86 general purpose I/O lines, 32 general purpose working registers, real time counter, six flexible timer/counters with compare modes, PWM, 4 USARTs, byte oriented 2-wire serial interface, 16-channel 10-bit A/D converter, and a JTAG interface for on-chip debugging. The device achieves a throughput of 16 MIPS at 16 MHz and operates between 4.5-5.5 volts.

By executing powerful instructions in a single clock cycle, the device achieves a throughput approaching 1 MIPS per MHz, balancing power consumption and processing speed with high-performance. [3]

Figure 2.1 ATMega 2560 Chip

2.1.3 Arduino Mega 2560

The Arduino Mega 2560 is an open-source microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 6 External Interrupts, 4 UARTs (hardware serial ports), one SPI, one I2C, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller. It's powered either by simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. [4]

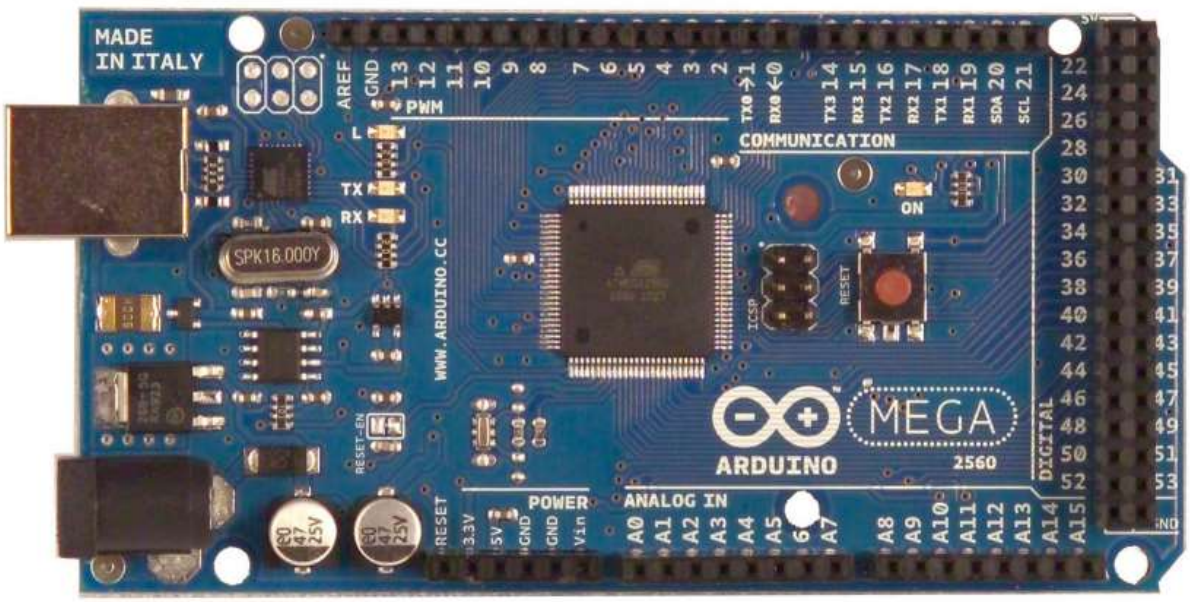


Figure 2.2 Arduino Mega 2560 Board

2.1.4 The Arduino History

The first Arduino was introduced in 2005. The project leaders sought to provide an inexpensive and easy way for hobbyists, students, and professionals to create devices that interact with their environment using sensors and actuators. Common examples for beginner hobbyists include simple robots, thermostats and motion detectors. Adafruit Industries estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands. [5]

2.2 Global Positioning System (GPS)

Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information. The system provides critical capabilities to military, civil, and commercial users around the world. The United States government created the system, maintains it, and makes it freely accessible to anyone with a GPS receiver. [5]

Since the GPS is the most effective tracking system with high accuracy, a lot of work has been done so far on using GPS as geographic information and navigational system for a mobile robot.

2.2.1 GPS History

The GPS was first originally designed for military and intelligence applications in the 1960s. After that, the US began the GPS project in 1973. The system of this project was developed by the U.S. Department of Defense (DoD), which basically used 24 satellites, and it became fully operational in 1995. During late 2005, the first of the next-generation of GPS satellites was launched. And today, GPS is used for dozens of navigation applications, route finding for drivers, map-making, earthquake research and climate studies. [6]

2.2.2 Principal of Operation

The basic idea of any satellite positioning system is to calculate the distance between a satellite and the user's current location. Every satellite transmits data that indicates its location and the time they sent out the signal to the GPS receiver. The GPS satellites are equipped with atomic clock on board to provide an accurate time reference. The distance between the particular satellite and the GPS receiver determined by calculating the travel time of a signal from the satellite to the receiver, where:

Travel time = signal reception time - signal transmission time
Distance = travel time x speed of light

Trilateration (triangulation) is used to calculate the current position of the GPS receiver based the information on GPS signal's travel time from three nearby satellites and their exact locations in the orbit. However, in order to determine one's location in 3D space, four satellites are needed instead of three.

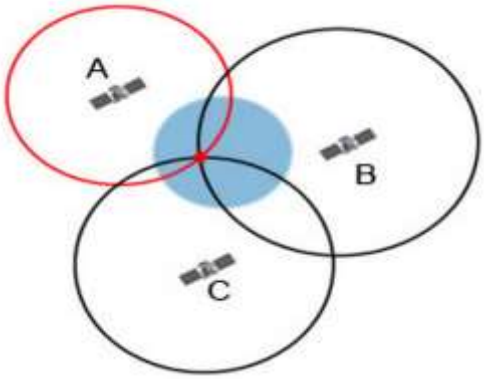


Figure 2.3 Trilateration

a. Coordinate System:

Basically, The Earth is divided into a grid of circular segments which are perpendicular to one another, called **latitude** and **longitude**. Latitude lines run horizontally, and are parallel to the equator. Degrees latitude are numbered from 0° to 90° north and south. Longitude lines run perpendicular to latitude lines. Degrees longitude are numbered from 0° to 180° east and west.

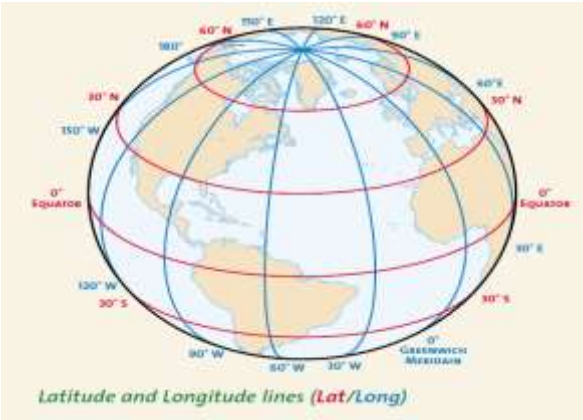


Figure 2.4 Latitude and Longitude Lines

2.2.3 Pmod-GPS

Pmod-GPS is an ideal solution for any embedded system in need of satellite positioning accuracy. It features the GlobalTop Gms-u1LP GPS antenna module which utilizes the Media-Tek-GPS-MT3329.

The Pmod-GPS utilizes a standard 6-pin connector and communicates via 2-wire UART at 5v level (TTL level). Also available on the board is a 2-pin connector for control of the NRST pin

to the module and also the RTCM pin for DGPS data of RTCM protocol (this feature is disabled by default, contact GlobalTop to enable).

Jumper	Pin	Description
J1	3DF	Indicates the GPS status
J1	RXD	Input-Serial Commands
J1	TXD	Output-Serial Data
J1	1PPS	One Pulse Per Second output synchronized with GPS time
J1	GND	Ground
J1	VCC3V3	Input voltage
J2	NRST	Active low reset
J2	RTCM	Inactive by default
J3	Coin Cell	Back-up Battery
J4	Plug-In	Used to plug an external GPS antenna

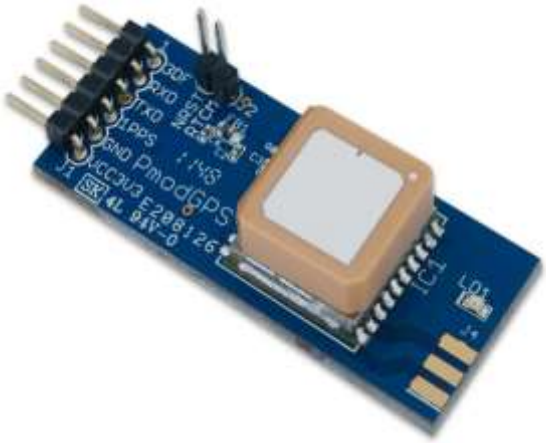
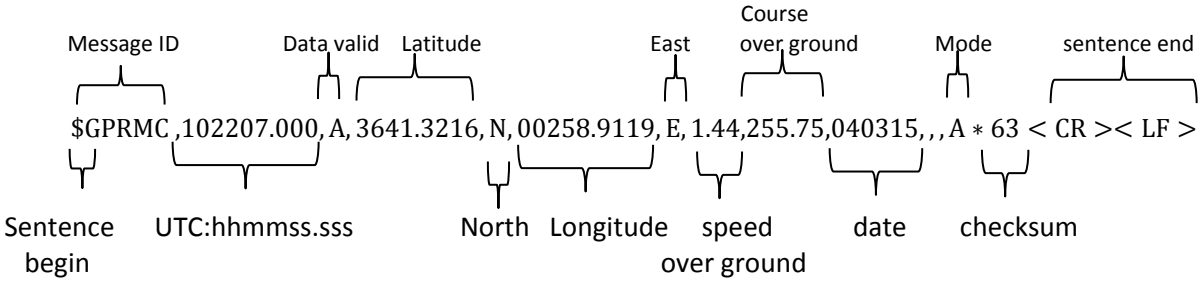


Figure 2.5 The Pmod-GPS

a. Interface Protocol of The Pmod-GPS Module:

The PmodGPS uses sentences based on National Marine Electronics Association (NMEA) protocols for data output. Each NMEA message begins with a ($) dollar sign. The next five characters are the talker ID and the arrival alarm. The PmodGPS talker ID is "GP" and the arrival alarm is the specific sentence output descriptor. Individual comma separated data fields follow these five characters. After the data fields there is an asterisk followed by a checksum. Each sentence should end with <CR><LF>.

To better understand this, we can see an example GPS sentence from our PmodGPS:



In order to find a location on a map by using the latitude and longitude of that location, we need to convert coordinates from degrees, minutes, and seconds format to decimal degrees format using the following formula:

Decimal value = Degrees + (Minutes/60) + (Seconds/3600)

2.3 Wi-Fi

Wi-Fi is a wireless networking technology that allows computers and other devices to communicate over a wireless signal. It describes network components that are based on one of the 802.11 standards developed by the IEEE and adopted by the Wi-Fi Alliance (802.11a\b\n\g).

Wi-Fi is the standard way computers connect to wireless networks. Most mobile devices, video game systems, and other standalone devices also support Wi-Fi, enabling them to connect to wireless networks as well. When a device establishes a Wi-Fi connection with a router, it can communicate with the router and other devices on the network. However, the router must be connected to the Internet (via a DSL or cable modem) in order to provide Internet access to connected devices. Therefore, it is possible to have a Wi-Fi connection, but no Internet access.

Since Wi-Fi is a wireless networking standard, any device with a "Wi-Fi Certified" wireless card should be recognized by any "Wi-Fi Certified" access point, and vice-versa. [7]

2.3.1 Socket

Sockets are the fundamental technology for programming software to communicate on TCP/IP networks. A socket provides a bidirectional communication endpoint for sending and receiving data with another socket. Socket connections normally run between two different devices on a LAN or across the Internet, but they can also be used for interprocess communication on a single computer.

Socket endpoints on TCP/IP networks each have a unique address that is the combination of an IP address and a TCP/IP port number. When creating a new socket, the socket library automatically generates a unique port number on that device, and the programmer can also specify their own port numbers in specific situations. [8]

a. Port Number:

A port is an internal address reserved for a specific application on a computer, a port can be either a TCP/UDP port, depending whether it’s linked to the TCP/UDP protocol at the transport layer, a port can be any number between 0-65535, frequently used TCP/IP applications are assigned port numbers under 1024 to avoid being mixed up with other applications; this ports also called well known ports.

2.3.2 Server/Client Structure

A server manages most processes and stores all data. A client requests specified data or processes. The server relays process output to the client. Clients sometimes handle processing, but require server data resources for completion. Because the Arduino mega is the provider of data, we have set the Arduino mega as a server, and our computer as a client.

a. Steps to set up Sever/Client (Arduino/PC) and TCP/IP Session:

Table 2.1 Illustrates how to set up a Server/Client structure:

ARDUINO: Server	**PC: Client**
Create a socket.	Create a socket.
Bind the socket to an address (the address is the ip address+ Specified TCP-Port).	Connect the socket to the address of the server (the address is the ip address+ Specified TCP-Port).
Accept a specified client (our PC).	Stay connected.
Send and receive data.	Send and receive data.

Table 2.1 Set up a Server/Client structure

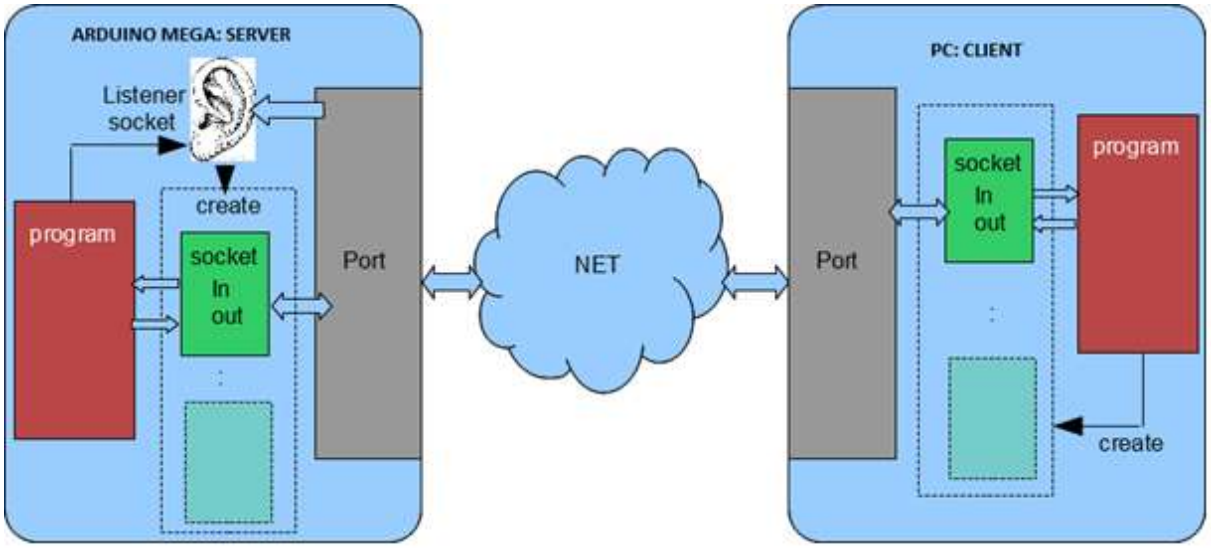


Figure 2.6 Server/Client Structure

2.3.3 Wi-Fi Module ESP8266

The ESP8266 is a low power consumption Wi-Fi module that utilizes a standard 8-pin connector and communicates via 2-wire UART at 3.3V Level. It supports 802.11 b/g/n standards,

Wi-Fi 2.4 GHz, WPA/WPA2 encryption. It has a TCP/IP stack and a user-programmable 32-bit microcontroller built on-board; this microcontroller is programmed using AT commands to add networking features to any microcontroller-board.

The ESP8266 is used in Smart power plugs, Home automation, Mesh network, Industrial wireless control, IP Cameras, Sensor networks, Wearable electronics, Wi-Fi location-aware devices, Security ID tags, and other electronic devices. So, the ESP82266 is a great solution to get a robot on a network or on the Internet.

Pin	Description
RXD	Input Commands
GPIO_0	Not Connected
GPIO_2	Not Connected
GND	Ground
VCC	3.3V input voltage
RST	Active low reset pin
TXD	Output Data

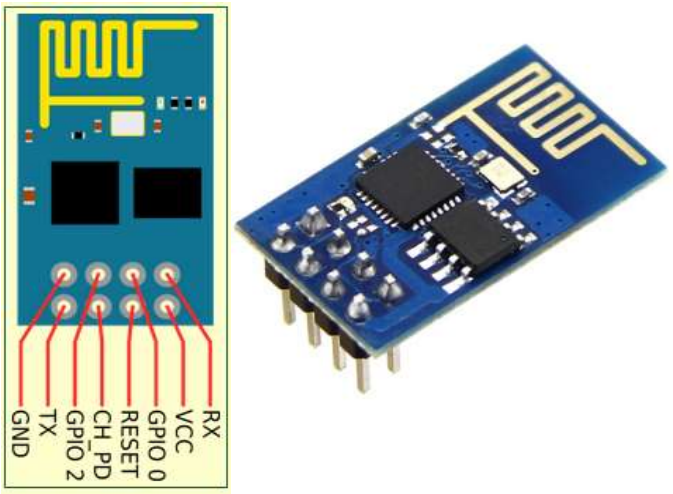


Figure 2.7 ESP8266 Chip

Figure 2.8 describes the internal structure of the ESP8266:

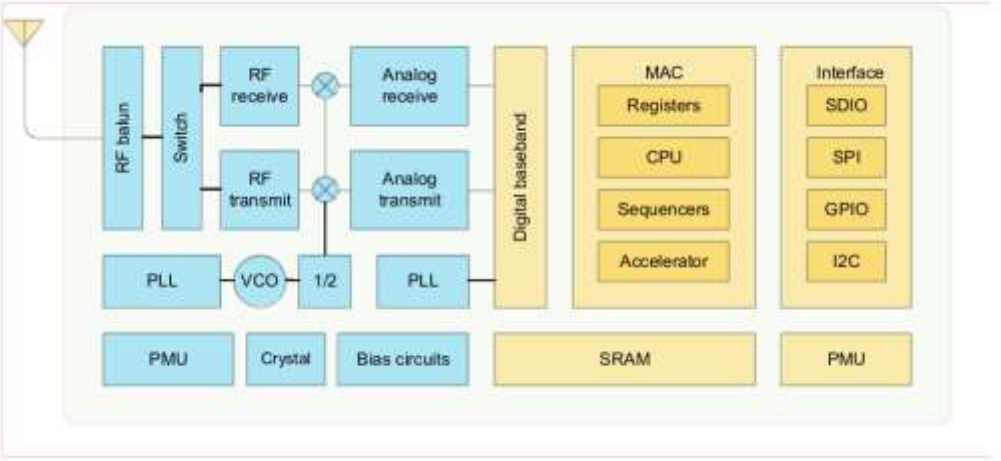


Figure 2.8 Internal Structure of the ESP8266

2.3.4 Logic Level Shifters (HEF4050BP)

Level shifters are used to shift down/up logic levels, the ESP8266 UART functions at 3.3V for logic high. The Arduino mega UARTs function at TTL level. So, the input signals to the ESP8266 have to be shifted down to 3V3 logic level, because if we don't do this, we are going to blow up our Wi-Fi module, so we need to shift those signals down. However, 3.3V logic level can be recognized by TTL level devices, that's why we are going to use only one level shifter.

There exists a bundle of level shifters; each one has its criteria of usage: voltage dividers circuits, NPN-Transistors based level shifter circuits, and non-inverting buffers. We have chosen the HEF4050BP buffer because it supports high baud-rates.

a. CMOS HEX BUFFER HEF4050BP:

The CD4050B device is non-inverting hex buffers, and feature logic level conversion using only one supply voltage (VDD). The input-signal logic level can be shifted down to 3.3V logic level at high baud-rate. It is powered as follows VDD=3.3V and VSS=GND.

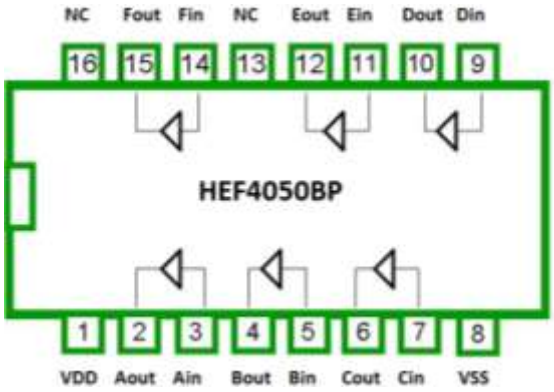


Figure 2.9 HEF4050BP Pin_Out

2.4 DC Motors:

The mobile robot system takes input voltage as actuator input, and outputs the rotational speed of the two wheels, the actuator most used for mobile robot is **DC motor** because their torque-speed characteristics are achievable with different electrical configurations, and their speeds can be smoothly controlled. Moreover, we added a gear system to provide our DC motors with a higher torque.

2.4.1 Motor Driver

Since DC motors require more current than typical microcontrollers such as Arduino can provide through their Input/output pins, we need some type of a switch that can accept a small

current, amplify it and generate a larger current, which further drives a motor .This entire process is done by what is known as a **motor driver.**

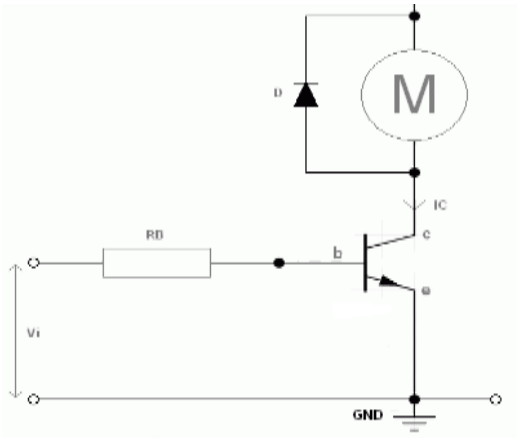
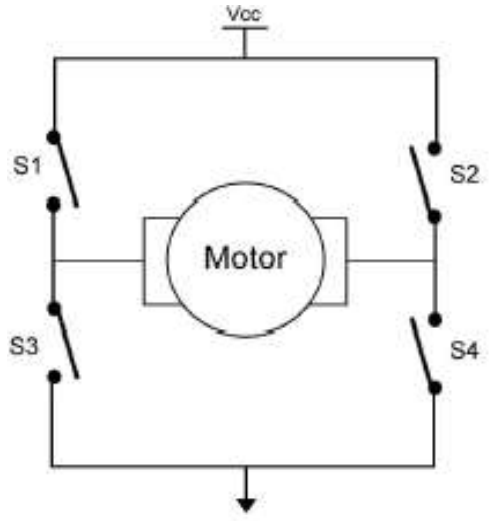


Figure 2.10 Driving motor using transistor as switch

a. The H-Bridge:

Standard solution to drive DC motors is to use an **H-bridge**, which is basically a set of transistors that can switch high currents by applying low-power input signals.

Turning a motor ON and OFF requires only one switch to control a single motor in a single direction, if we want to reverse the direction of the motor we need to reverse its polarity. This can be achieved by using four switches that are arranged in an intelligent manner such that the circuit not only drives the motor, but also controls its direction.



S1	S2	S3	S4	
1	0	0	1	Forward
0	1	1	0	Backward
1	1	0	0	Brake
0	0	1	1	Brake
0	1	0	1	Short Circuit
1	0	1	0	Short Circuit

Figure 2.11 H-Bridge circuit for DC motor control

Since H-bridges are used in many applications, complete chips already exist that suit our purpose; we have chosen to work with **the L293D.**

2.4.2 Motor Driver IC L293D

The L293D is a dual H-bridge motor driver IC that can drive two motor simultaneously. The L293 is designed to provide bidirectional drive currents of up to 600 mA at voltages from 4.5 V to 36 V. It is a 16 pin IC having two enables pins which should always be remain high to enable both the H-bridges.

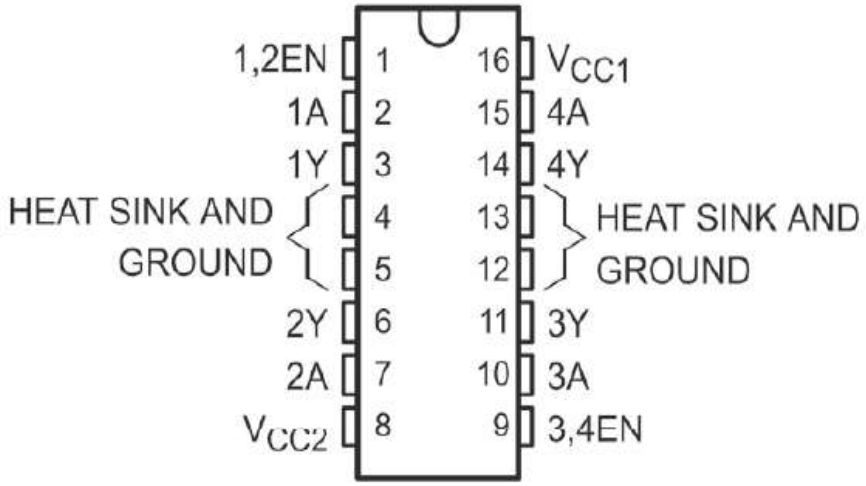


Figure 2.12 L293D Pin_Out

2.5 Mechanical Description

Mobile robots use several classification of wheels situation such as differentially driven, Car-type, Omni-directional, and Synchro drive. These wheels configurations need different controlling techniques in motion and trajectory.

2.5.1 Differential Drive Robot

A design frequently used for computer-controlled mobile robots is the **differential drive**, which consists of two drive wheels, each with its own controlled motor, and one free-wheeling castor for balancing robot's structure. And the way these robots work is by controlling the relative rate of rotation of its wheels. So, forward motion is achieved when both wheels are driven at the same rate, turning right is achieved by driving the left wheel at a higher rate than the right wheel and vice-versa for turning left. Moreover, this type of mobile robot can turn on the spot by driving one wheel forward and second wheel in opposite direction at the same rate.

2.5.2 Mechanical Design

The brunt of the mechanical design was done by building everything from scratch without any pre-assembled parts. We essentially cut the Plexiglas of the platform base and constructed the system for everything to work together.

Our mobile robot belongs to the category of a car-like robot (CLR) based on the differential drive system. It is mounted on a well-designed chassis that has a length of 30.6 cm and a width of 20.5 cm. The rear wheels hacked from an old toy which are 6cm in diameter, whereas the front castor wheel is 4 cm.

a. Chassis Design

While designing the chassis for our robot, we took in consideration to make it flexible and easy to be controlled; this chassis is designed using AutoCAD.

CHAPTER III

Hardware System Design

In this chapter, the hardware system design is shown step by step; each part of the system is explained alone then the overall system design is presented at the end of the chapter.

3.1 Introduction

First, this chapter introduces the UART communication protocol and the Pulse Width Modulation (PWM). Then, it describes the hardware system design, the system is designed using iterative technique; where each important functionality is designed and realized alone. And finally it demonstrates the overall system design.

3.2 Universal Asynchronous Receiver/Transmitter UART

Serial communication is the main method used for communication between external microcontrollers and computers, and between microcontrollers themselves and their shields. The Universal asynchronous receiver/transmitter (or UART) is a type of asynchronous receiver/transmitter, UARTs devices use no clock between them for data synchronization, but uses internal clocks with the same frequency to synchronize data transfer and validate data. The main wires in UART communication are two data transmitting lines which are: a Transmit Data line **TxD**, and a Receive Data line **RxD**. Devices send data over the TxD line which is an output, and receive data over the RxD line which is an input. A UART transmits data in chunks often called words; each word contains a Start bit, data bits, an optional parity bit, and one or more Stop bits. Asynchronous serial interfaces are widely used because they are cheap, and easy to use.

a. Communication Protocol

A protocol is a set of rules that defines how devices manage communications; UART communication protocol defines how the bits travel, including the bit rate, in what order the bits transmit, how many bits in a single data transfer, and some error detecting bits.

- Bit Rate or Baud Rate: the speed in which bits are transmitted. Each bit has fixed time duration determined by the transmission rate (300, 1200, 2400, 4800, 9600, 19.2K, 38.4k, 57.6k, 115.2k).
- Start bit: marks the beginning of a new word: When detected, the receiver synchronizes with the new data stream.
- Number of Data bits: determines the number of bits per word (5-9 bits).
- LSB or MSB first: The least significant bit is sent first or the most significant bit sent first.
- The parity bit: is added to make the number of 1's even (even parity) or odd (odd parity), this bit can be used by the receiver to check for transmission errors even, odd, none, mark, space).
- Stop bit: marks the end of transmission, Receiver checks to make sure it is '1', Separates one word from the start bit of the next word (1, 1.5, 2).

Figure 3.1 UART Bits Description

b. UART Options in Arduino Mega 2560

Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

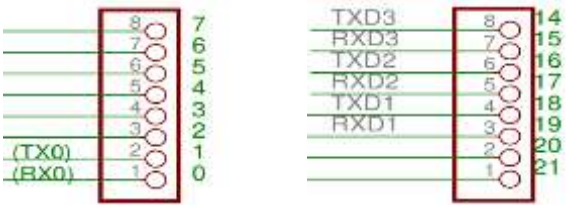


Figure 3.2 Arduino RX_TX pins

3.3 Pulse Width Modulation (PWM)

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, we change, or modulate that pulse width.

The **duty cycle**, D, refers to the percentage of the period for which the signal is on. The duty cycle can be anywhere from 0, the signal is always off, to 1, where the signal is constantly on. A 50% D results in a perfect square wave.

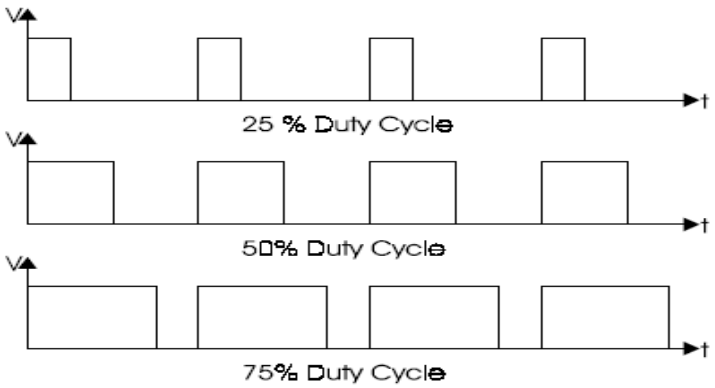


Figure 3.3 Duty Cycles

a. Controlling DC Motors With PWM

The speed controller for a DC motor works by varying the average voltage supplied to the motor. PWM is one such speed controller where we can get varying voltage according to the duty cycle of the PWM output signal. Rather than simply adjusting the voltage sent to the motor, we can switch the motor supply on and off where switching is done so much fast that the motor only notices the average effect.

If switching is done fast enough the motor does not have time to change speed and it gives an almost constant speed. Thus the speed is set by Pulse Width Modulation (PWM).

3.4 Description Of The System

Each functionality in robot contains Arduino Mega as a master of the design. So, the main parts are: Arduino Mega and GPS (PmodGPS), Arduino Mega and Wi-Fi module (ESP8266), Arduino Mega and the two DC Motors.

3.4.1 Arduino Mega and GPS (Pmod GPS)

This part is about interfacing the Arduino Mega with the GPS Module which provides coordinates information including precise time, date, heading, speed, and other useful information related to the specific satellites in view.

We used The Pmod GPS which operates on an input voltage between 3v and 3.6v, so we connected its **Vcc** to the Arduino **3.3v** pin. Then, since the Pmod GPS uses UART protocol for data transmission and reception, its **RXD** and **TXD** are connected to **TX2** (pin16) and **RX2** (pin17) on the Arduino, respectively. The interface is programmed to use a baud rate of 9600, 8 data bits, with no parity, and a single stop bit.

The PmodGPS's **3DF** pin indicates the status of the user's positional fix and drives the fix LED (LD1). When the module has a constant fix (2D or 3D) this pin stays low, so we used the digital pin **41** on the Arduino to determine fix status.

The **1PPS** pin on J1 provides a one pulse-per-second output synchronized with GPS time using digital pin **42** on the Arduino.

Finally, the reset pin **(NRST)** on J2 that allows normal operation in active low, is connected to digital pin **40** on the Arduino, if we toggle the NRST pin it will completely reset the module.

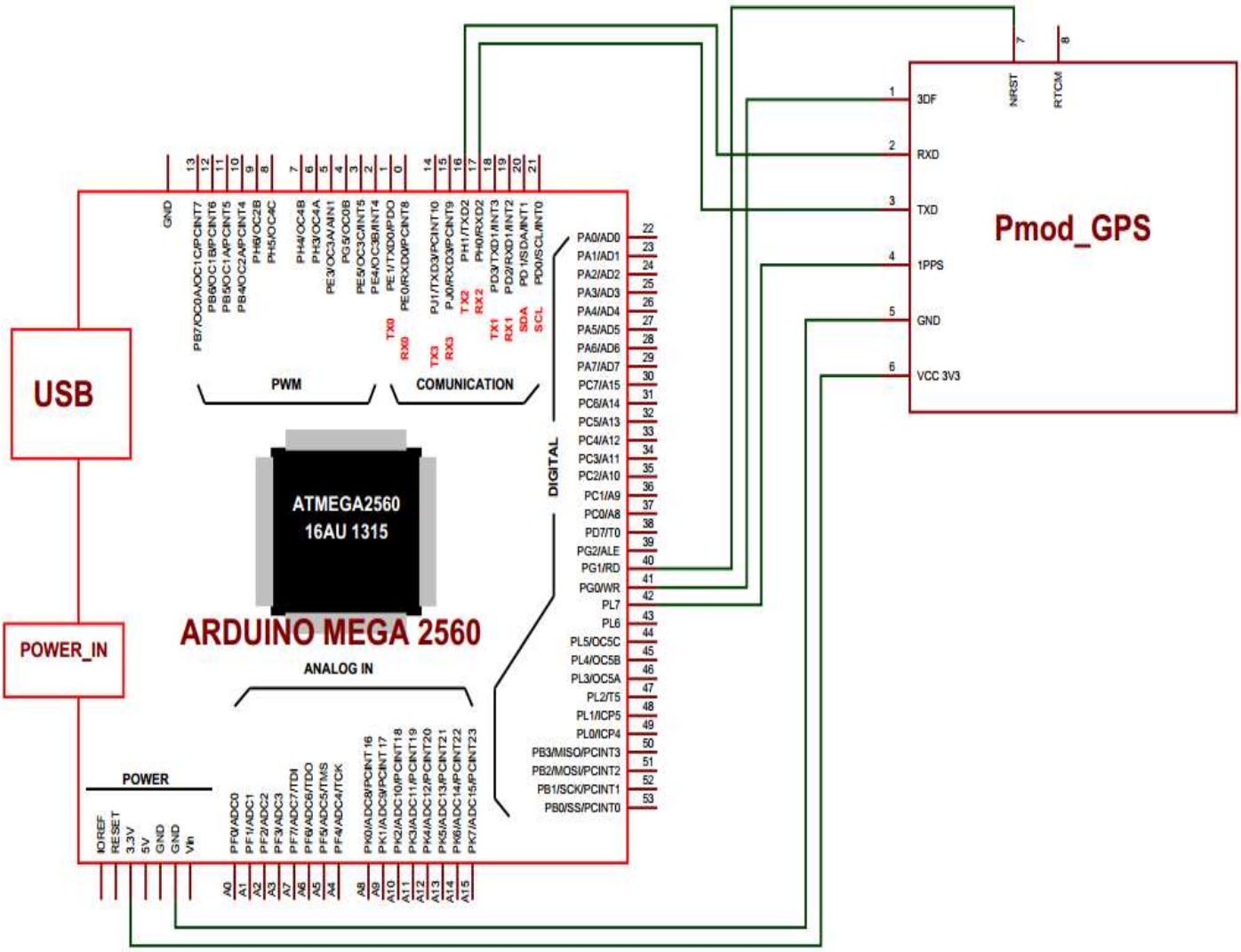


Figure 3.4 Interfacing PmodGPS with Arduino Mega

3.4.2 Arduino Mega and Wi-Fi Module (ESP8266)

In this part we are going the interface the Wi-Fi Module in order to add the Wi-Fi functionality to the Arduino and make it able to connect wirelessly with other devices.

We used The ESP8266 Wi-Fi module which communicates via a UART serial connection. So first, we powered the ESP8266 by connecting the v_{cc} and **GND** pins directly to the **3.3v** and **GND** pins of the Arduino Mega.

However, the ESP8266 needs to communicate serially at **3.3v level**, since The Arduino Mega UARTs function at TTL level (**5v**), we shifted down the input signals to the ESP8266 to 3.3v using a Level shifter (HEF4050BP). The HEF4050BP is a CMOS hex buffers that is featured with a

logic level conversion using only one supply voltage v_{DD} which is connected to the **3.3v** of the Arduino Mega along with the **CH_PO** pin of the ESP8266 via a 10kΩ pull up resistor. The Arduino **TX1** on pin18 is connected to the input pin **3 (A)** of the Level Shifter (HEF4050BP) that accepts an input voltage at 5v. Then, the voltage is shifted to 3.3v through the output pin **2 (G=A)** which is connected to the **RXD** pin of the ESP8266. While the ESP8266's **TXD** pin is connected directly to the Arduino **RX1** (pin 19).

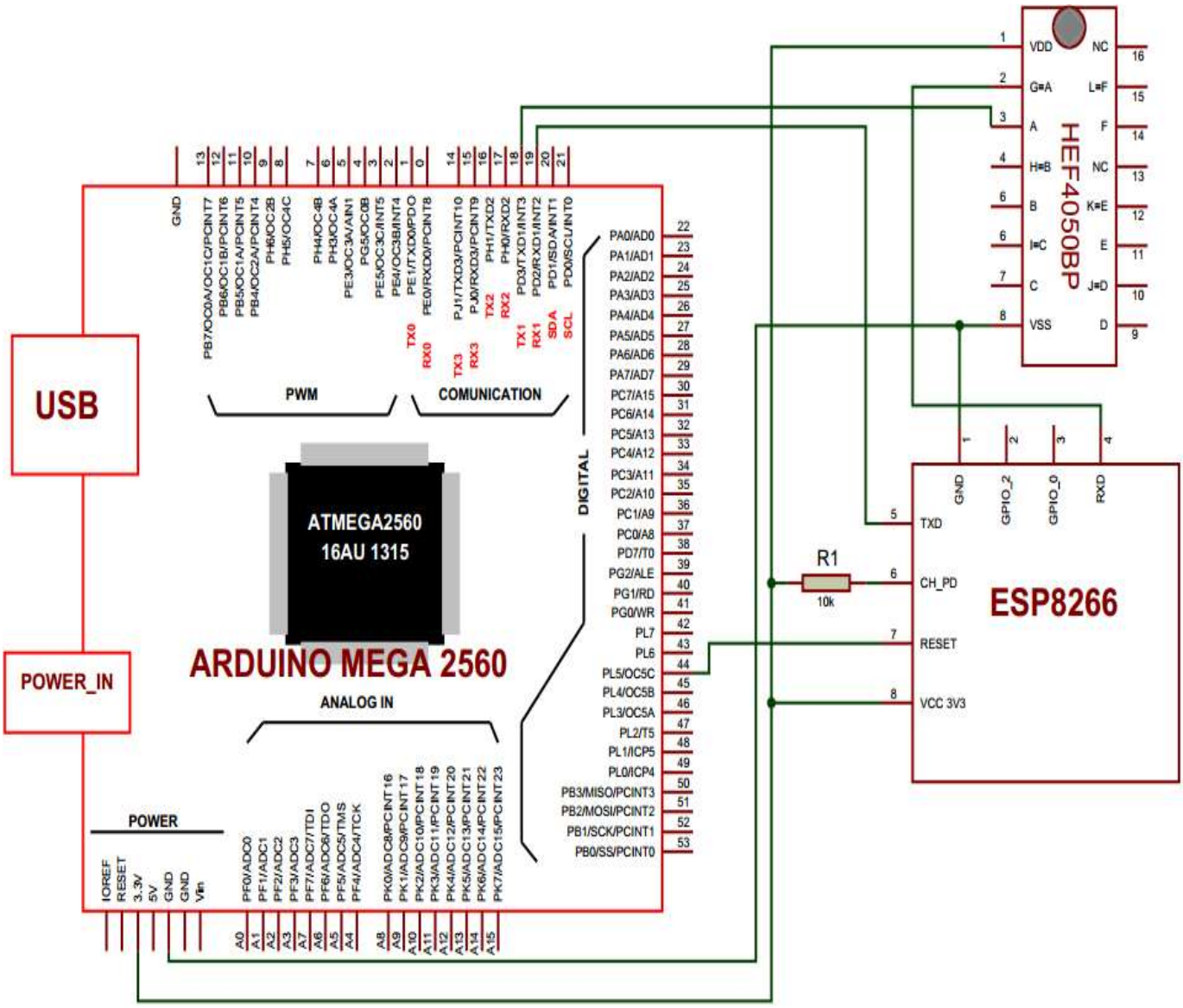


Figure 3.5 Interfacing ESP8266 with Arduino Mega

3.4.3 Arduino Mega and DC Motors:

In this part we are going to control the DC motors; this system includes using the motor driver (L293D).

Since the L293D is a dual H-bridge motor driver IC that can drive two DC motors simultaneously, we configured the L293D in a specific way that will allow us to control motors easily.

The L293D has two $\boldsymbol{V_{cc}}$ pins (**8** and **16**). The pin $\boldsymbol{V_{cc2}}$ (8) provides the power for the motors, and $\boldsymbol{V_{cc1}}$ (**16**) for the chip's logic. We have connected $\mathbf{V_{cc1}}$ to the Arduino 5V pin. However for$\mathbf{V_{cc2}}$, we should provide them with an external power supply (9v) using pin **8** connected to the positive power supply, and the ground is connected to the ground of the Arduino.

The L293 current driver chip amplifies the input signal received on pins **2, 7**, **10** and **15** to outputs pins **3**, **6**, **11** and **14**. This allows us to take a low power signal from our circuit and transform it into a higher power signal for the motors. Using these 4 input/output the L293 can drive two motors forward and backward.

We connected the left motor to pins **3** and **6**, and the right motor to pins **11** and **14**.

In order to drive the left motor Forward/Back and Left/Right, we generated PWM signals from the Arduino on **pin2** and **pin7**. We can use the same technique to drive the right motor with pins **10** and **15**.

The speed of the DC motors can be controlled by sending PWM signals to the L293 “enable input” pins. **Pin1** drives the speed of the left motor and **pin9** drives the speed of the right motor.

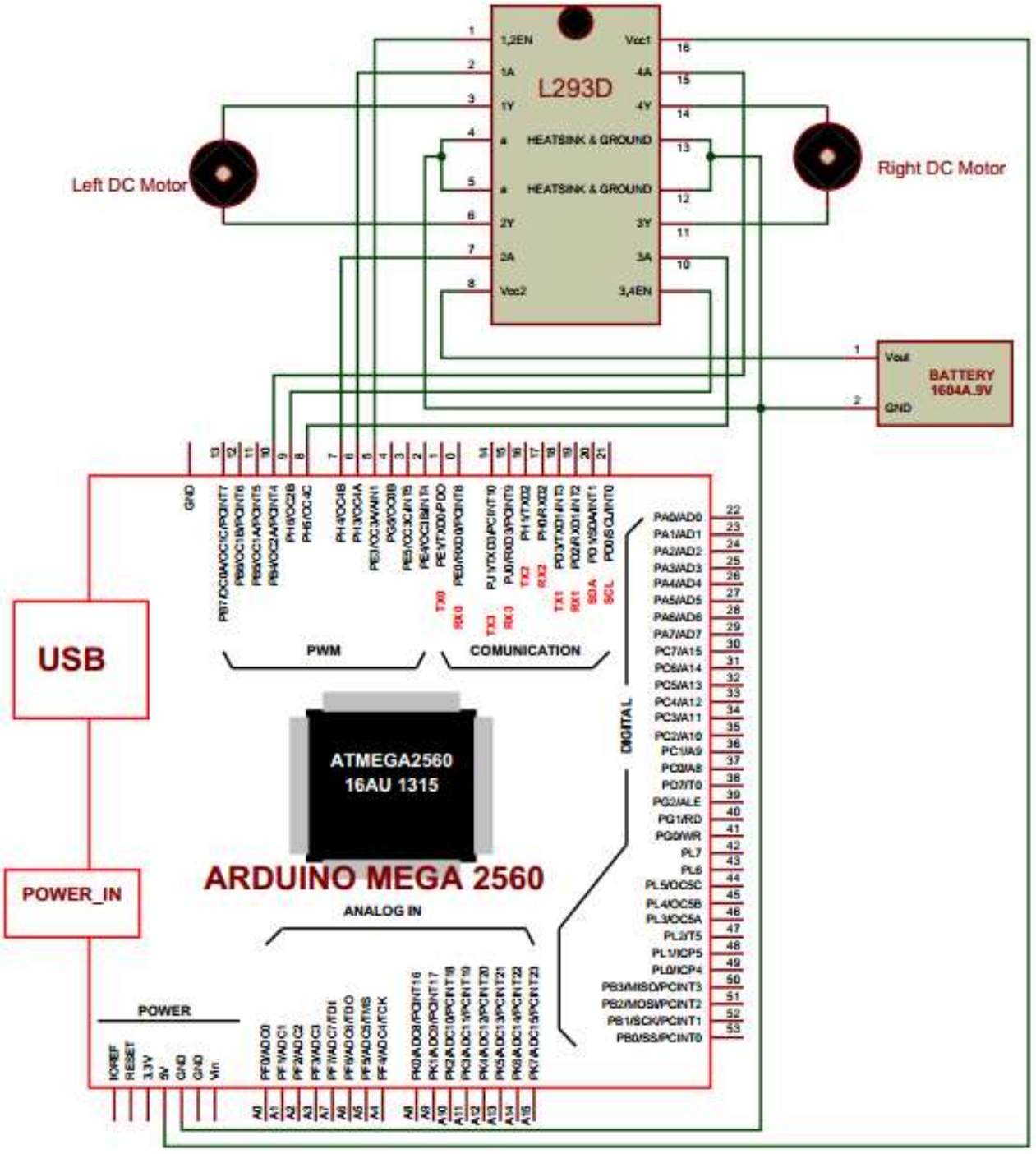


Figure 3.6 Interfacing DC motor & L293D with Arduino Mega

3.5 Overall System Design

The overall system connects all the parts together; controlling the DC motor, Receiving the GPS data; Setting up the Wi-Fi connection.

As shown in the design, we added a 3.3v external power supply for the PmodGPS, since supplying it from the Arduino (3.3v) along with the ESP8266 is not enough for them to operate together.

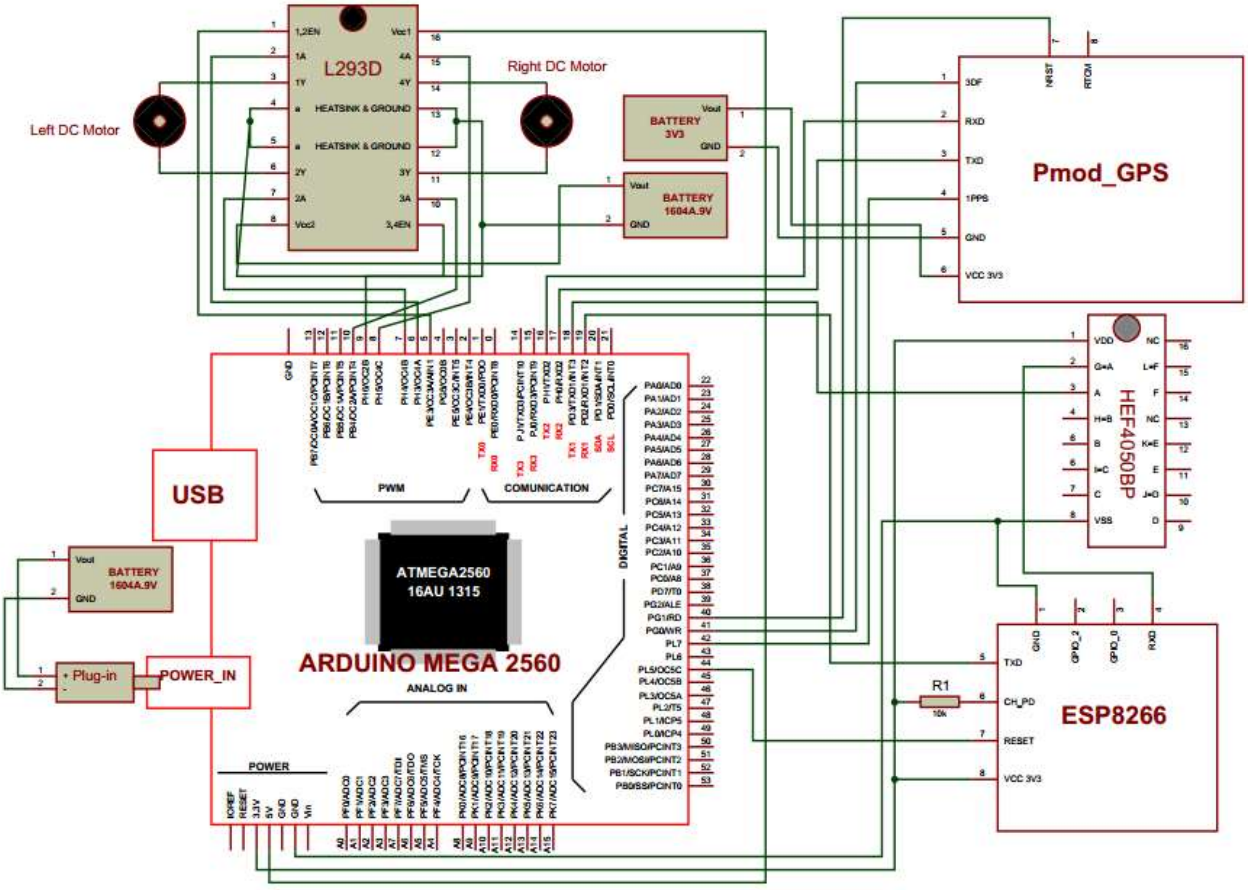


Figure 3.7 Overall System Design

Figure 3.8 shows a top view picture of the robot and its connections:

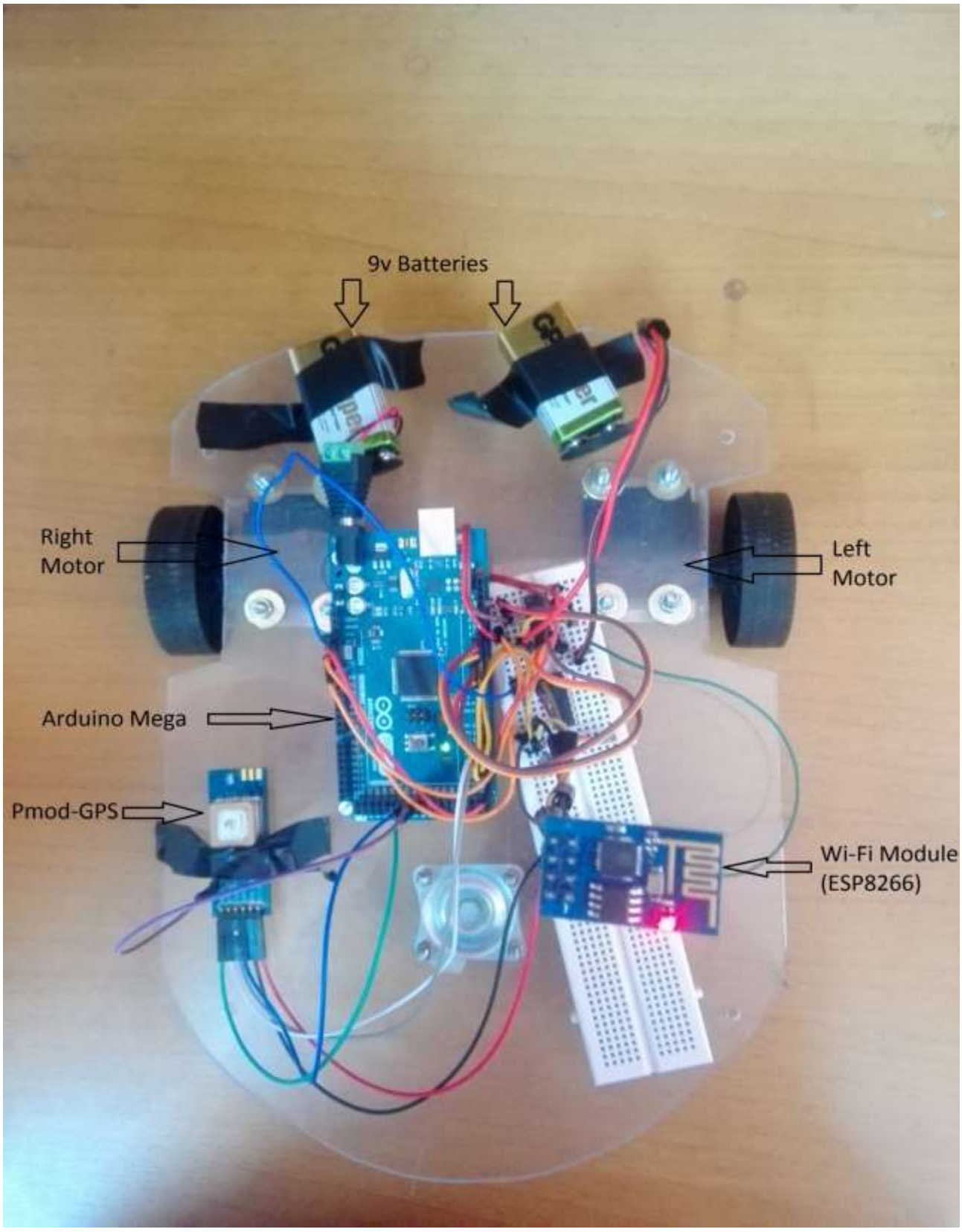


Figure 3.8 A top view picture of the robot

CHAPTER IV

Software System Design

This chapter defines the programming languages and the environments used in the project. Then it presents the software design process and flowcharts of the algorithm used to program the Arduino and the Visual C# application.

.

4.1 Programming Language and Environment

Visual C# and Arduino are the programming languages that are used as the interface languages in this project. The programming environments that are used to write visual C# program is Microsoft Visual Studio Express 2013, while the Arduino programming was developed using the Arduino Interface Development Environment.

4.1.1 Arduino Programming Language

Arduino programming language is a simplified version of C or C++ programming language, this programming language does not need to include the header file at the beginning of the coding. Besides that, there are only two functions are needed in order to make a cyclic executive program, these functions are setup () and loop (). Setup () is the function that is used to initialize the settings and only runs once at the beginning of the program. While the loop () is the function that is called continuously.

4.1.2 Visual C# (C Sharp) Programming Language

C# is a relatively new language that was unveiled to the world when Microsoft announced the first version of its .NET Framework in July 2000. C# is designed to be fully compatible with Microsoft's .NET initiative while taking advantage of powerful aspects from C, C++ and Java. Its syntax is similar to C and C++ syntax. C# is simple, powerful, type-safe, and object-oriented programming language.

Visual C# is an implementation of the C# language by Microsoft. Visual Studio supports Visual C# with a full-featured code editor, compiler, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The .NET Framework class library provides access to many operating system services and other useful, well-designed classes that speed up the development cycle significantly.

4.1.3 Arduino Integrated Development Environment

Arduino IDE is an environment that consists of a message area, a simple text editor and menus. It allows the user to create “sketches” in order to let the Arduino board interacts with the device which connected to. The sketches that written by the user can be stored in sketchbook.

Besides writing the program, it also allows user to compile and upload the sketches to the Arduino board.

4.1.4 Microsoft Visual Studio Express 2013

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft which was released on April 2013. This integrated development Environment shell that was written using Windows Presentation Foundation (WPF) can support multiple document windows and floating tool windows.

Besides that, Microsoft Visual Studio can be used to develop console and graphical user interface (GUI) application along with Windows Forms applications, web sites, web applications, and web services.

4.2 Mapping

The Map form in the GUI contains only a single Web browser. This form accepts the latitude and longitude as arguments and uses this data to display the current position on the **Google Maps**. A query string is formed using round the latitude and longitude is passed to the web browser to navigate to the location which indicated by the latitude and longitude.

4.2.1 Google Maps JavaScript API v3

Google launched the Google Maps API (**A**pplication **P**rogramming **I**nterface) in June 2005 to allow developers to integrate and embed freely Google Maps into their websites, which offers services for retrieving static map images, and web services for performing geocoding, generating driving directions, and obtaining elevation profiles. In Google Map webpage; HTML is used to define the content of the web page, CSS is used to specify the layout of the web page, and JavaScript is used to program the behavior of the web page. Over 1,000,000 web sites use the Google Maps API, making it the most heavily used web application development API. So, this API allows us to locate our robot precisely.

4.3 Arduino Programming Development

4.3.1 PmodGPS

As we mentioned in chapter II, the PmodGPS receives data using sentences based on NMEA protocols. In this part we wrote a program that allows the GPS receiver to get a fixed position and then it sends the information to a Client (PC) via Wi-Fi.

When the GPS got a fix, the Arduino parses the received sentences from the GPS, then constructs another sentence which contains a Fix character, latitude, longitude, altitude, number of satellites used in data acquisition, speed in knots, speed in Km, and the current time, this buffer is sent to user through the network.

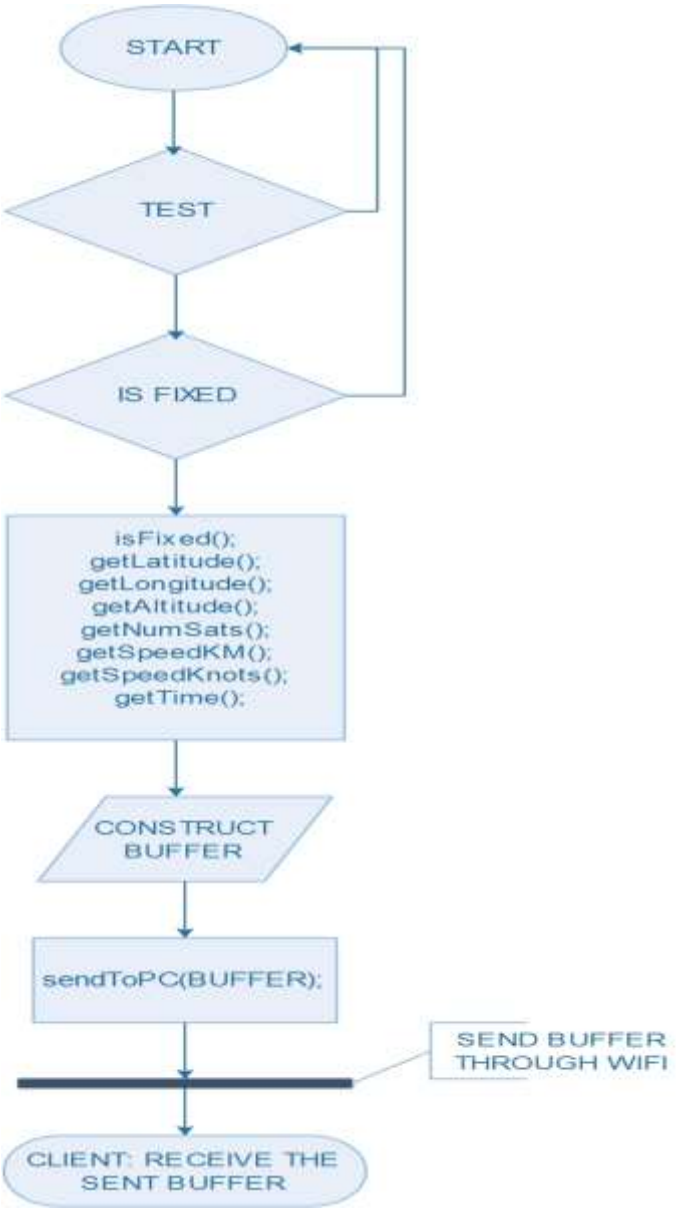


Figure 4.1 Flowchart for PmodGPS receiver

4.3.2 Wi-Fi Module ESP8266

First, we configured ESP8266 to log on to our Wi-Fi network (SSID and password are given as macro defines at the beginning), then we set ESP8266 as socket server with port number 8888, and it listens to incoming request.

In order to set the Mobile Robot as a server, the Arduino sends a specified sequence of AT commands to configure the Wi-Fi module in a way that allows Arduino to become a server.

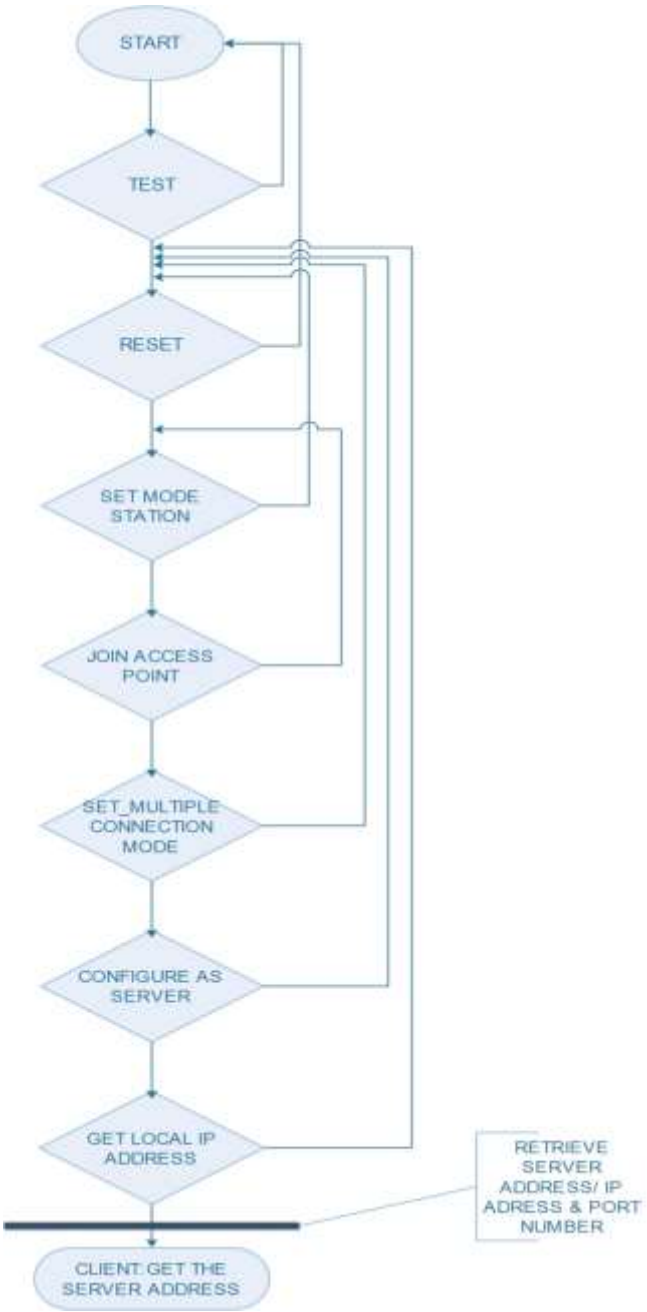


Figure 4.2 Flowchart for Wi-Fi Module ESP8266

4.3.3 Control DC Motors (Move Robot)

In the flowchart below represented by the Figure 4.3 shows how the Arduino receives a buffer from the client; this buffer represents a command that specifies the speed and the direction for the robot at that time, then these commands are converted to the corresponding PWM signals which are fed to the DC motors through the Dual H-bridge. The Dual H-bridge provides the needed current in order to rotate the DC motors effectively.

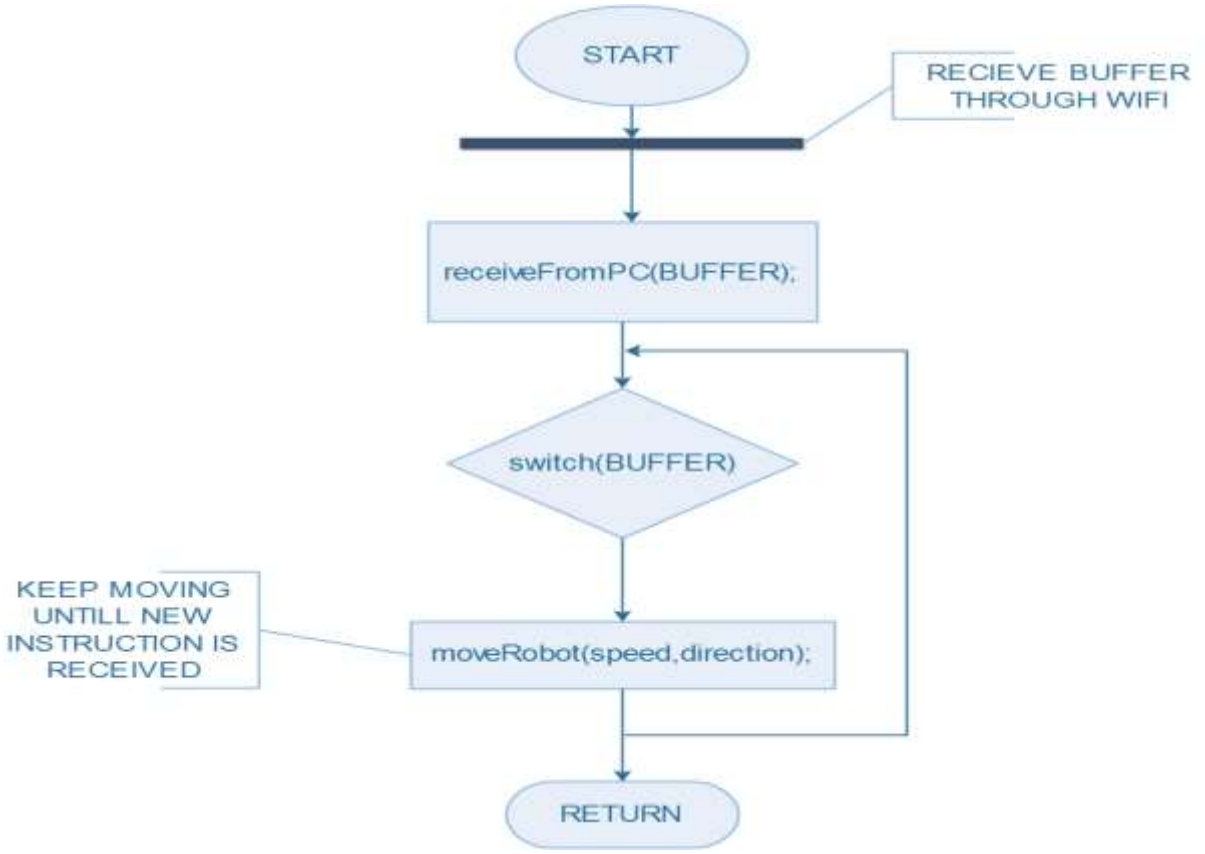


Figure 4.3 Flowchart to Move Robot

4.3.4 Arduino Overall Program

In the flowchart below represented by the Figure 4.4 shows how we set Arduino mega as a server, then the server sends GPS data to the client and receives commands for the motion control from the client.

The process of setting Arduino as a server is explained in the flowchart represented by the Figure 4.2. After this process the robot sends the constructed buffer from the GPS data which is explained in flowchart represented by Figure 4.1. And the Robot moves according to the specified speed and direction received from the user, this last process is explained in the flowchart represented by the Figure 4.3.

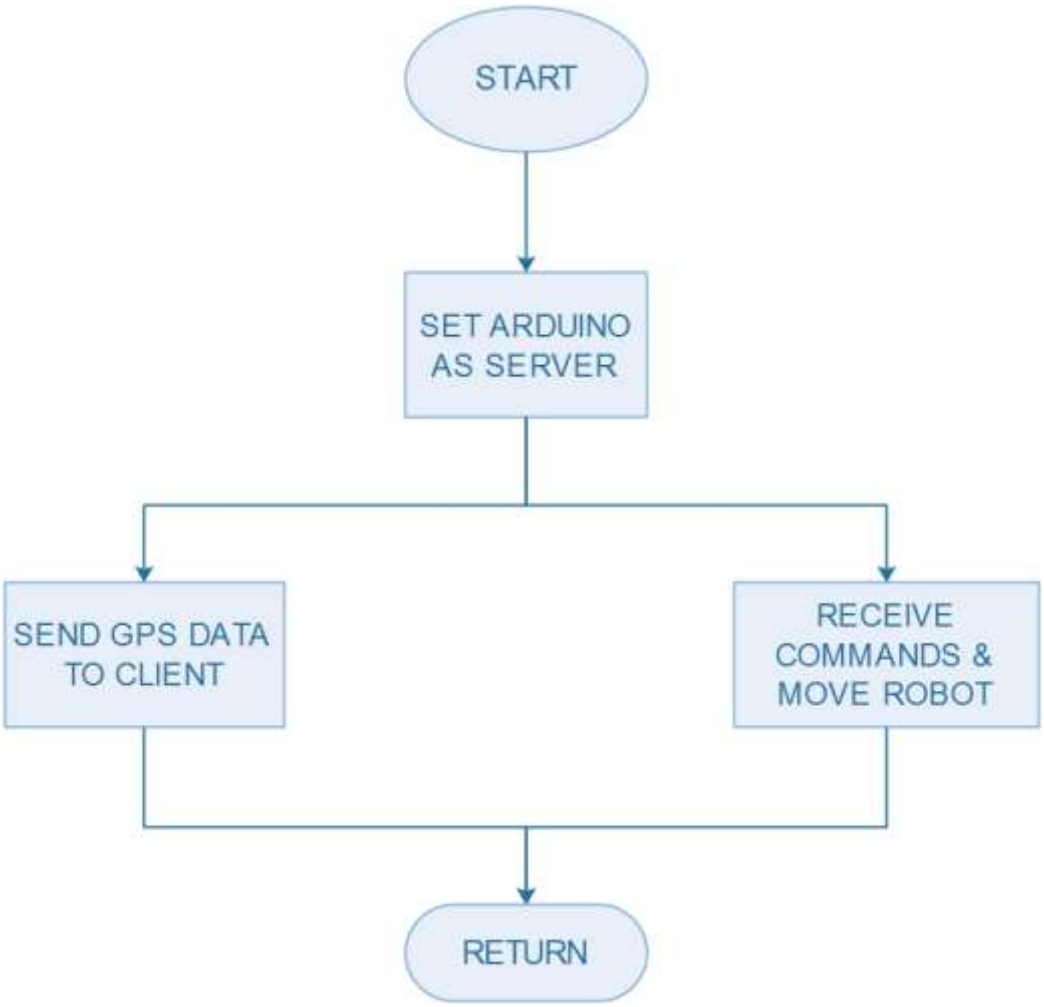


Figure 4.4 Flowchart of Arduino Overall Program

4.4 Visual C# (C Sharp) Programming Development

4.4.1 Set PC as Client

The flowchart below represented by the Figure 4.5 shows how we set up our computer as a client. First of all, we connect our computer to the Wi-Fi network, and then we enter the server address in the corresponding GUI field, the server address is gotten from Arduino serial monitor during setting Arduino as the server. Then if the client is connected, it can exchange data and commands with the server. The data is the GPS information which is the sent buffer as the flowchart in the Figure 4.5 shows, and the commands specifies the direction and speed to the Robot.

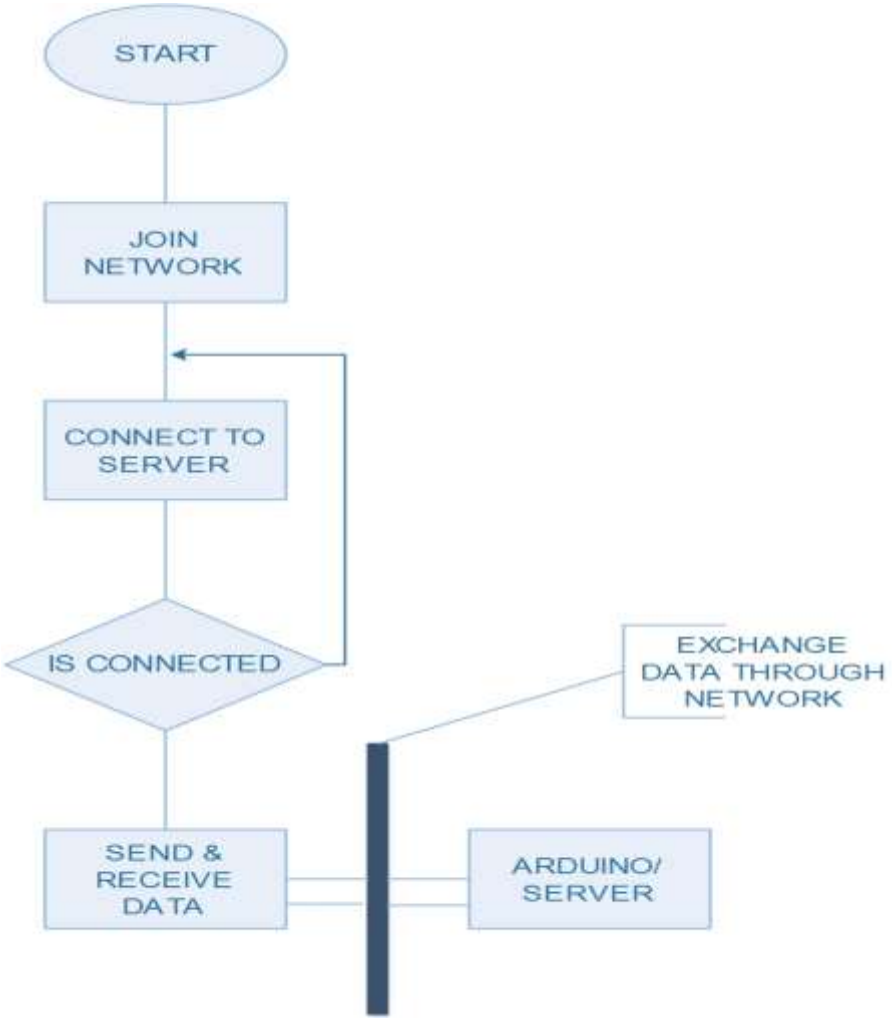


Figure 4.5 Flowchart to Set PC as Client

4.4.2 PC Robot Control

The flowchart below represented by the Figure 4.6 shows how to use the Graphical User Interface. The user can specify the speed and the direction for the robot, when the speed and direction are specified the GUI sends a specific command to the robot through the network, this command contains the speed and direction for the two DC motors.

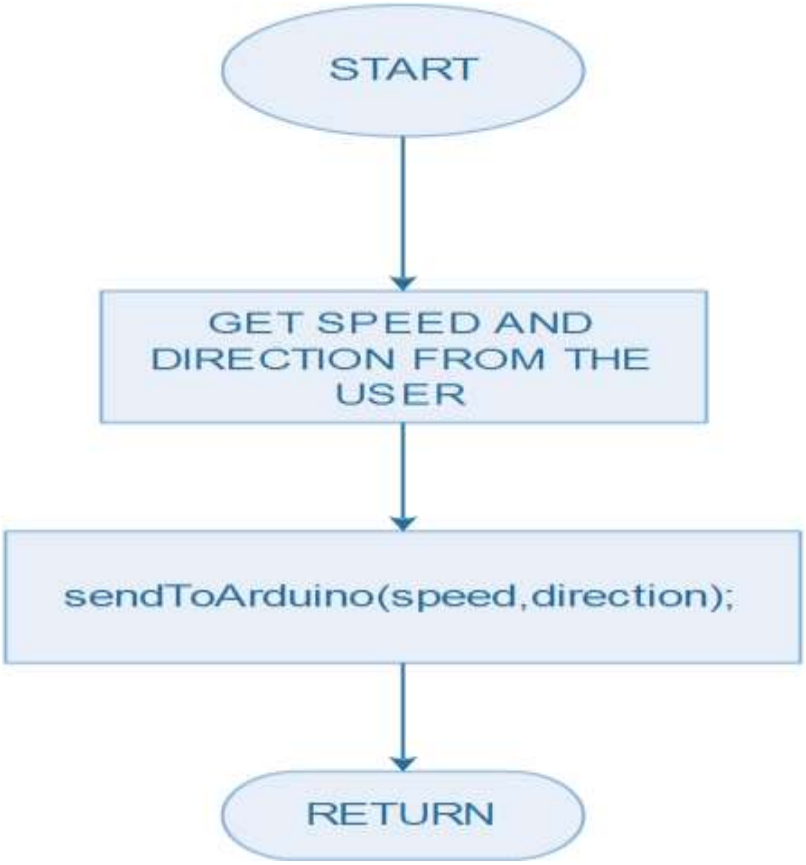


Figure 4.6 Flowchart of PC Robot Control

4.4.3 Client Parsing GPS Data

After the server has sent the buffer which contains the needed GPS data, this buffer is received and processed by the client. Then the decimal coordinates are fed to the Google Maps to locate the robot, and all the parsed data are displayed on the GUI in their corresponding fields.

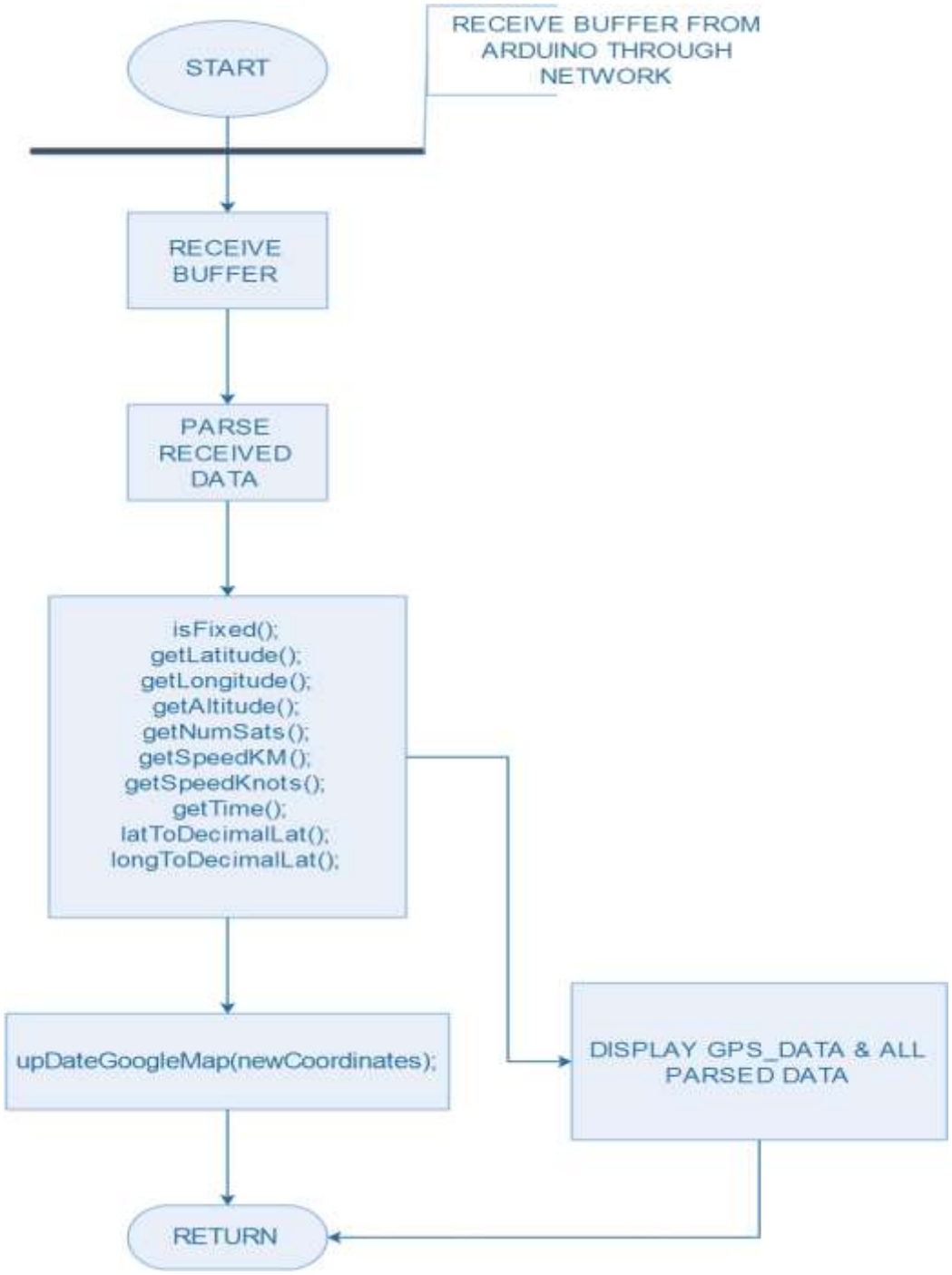


Figure 4.7 Flowchart of Client parsing GPS data

4.4.4 PC Overall Program

The flowchart below represented by the **Figure 4.8** shows how the GUI sets our computer as a server, this is done as the flowchart in the Figure 4.5 shows. Then the GUI receives and processes the GPS data, this is done by parsing the received sentences from the robot server through the network. The sentences are comma separated messages, the items between any two successive commas have specified order as mentioned and explained by the Figure 4.7. These items are displayed on the GUI, each item has a field where to be displayed.

The client also sends commands for the robot motion through the network, the usage of these commands by the robot is explained by the Figure 4.6.

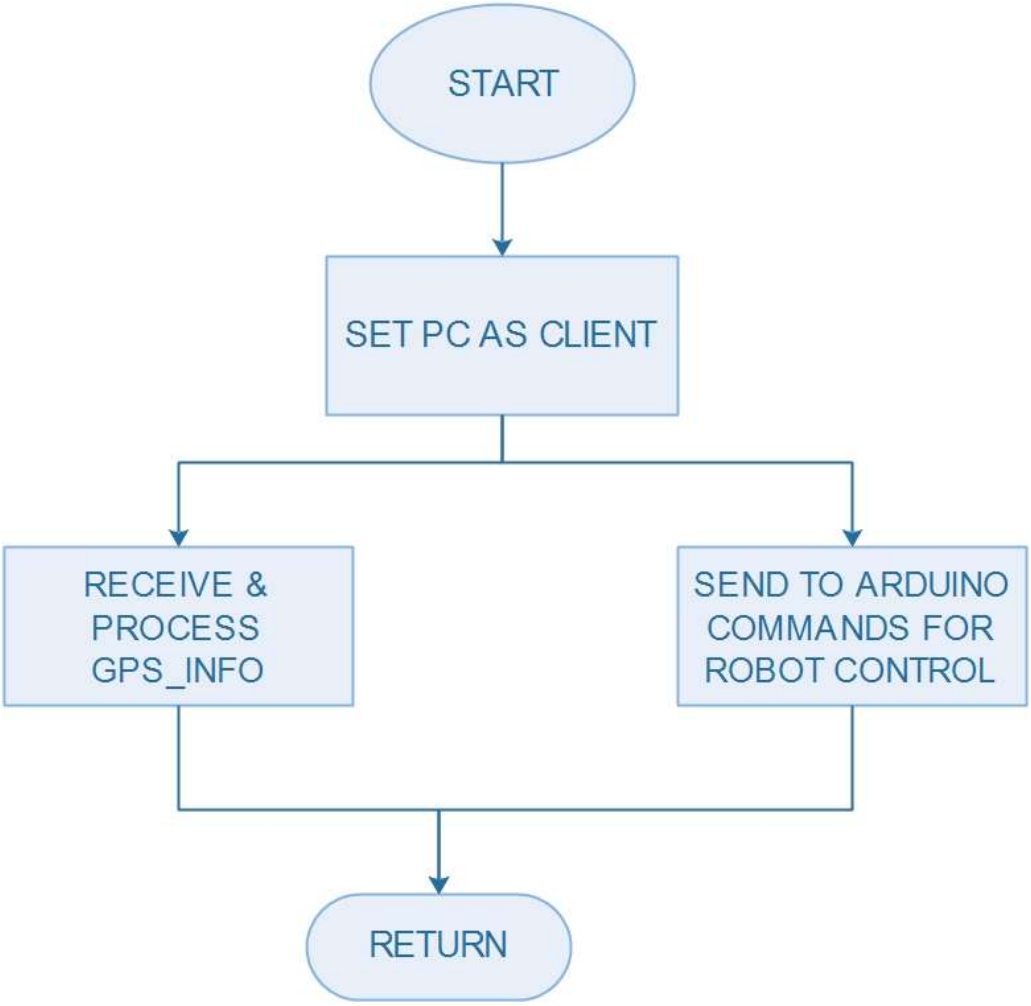


Figure 4.8 Flowchart of PC OVERALL PROGRAM

4.5 Final GUI

The GUIs shown below in **Figure 4.9** and **Figure 4.10** contain mainly five group lists: Net Settings, Robot Control, Google Maps, Current Location, and GPS Info, and also a Google Map container. In Net Settings, we enter the server address (Ip address + Port) in order to connect to the Arduino server. The robot motion is controlled using Robot control group list; this group contains two speeds, four directions, and a stop button. In Google Maps, we can locate our robot on a map; we can choose the map type: Satellite, RoadMap, Terrain, or Hybrid. In Current Location, the GUI displays the Decimal Coordinates, the Altitude, and the Degrees, Minutes, Seconds coordinates format. In GPS Info, we display other data like Date, Speed, and number of satellites used in GPS data acquisition, and also the buffer received from the server is displayed as a complete message.

Example of how to use this GUI, first of all we enter the server address, then we receive a message notifying that we are connected to the server. After that we can choose the speed and direction for robot from the Robot Control group list. Then we can locate our Robot on the Map by choosing the Map Type then clicking on Refresh Map button, on the map there is a Message-Marker which shows the current location of the Robot. Finally, we can notice the change in all the fields due to the updating values received from the Robot.

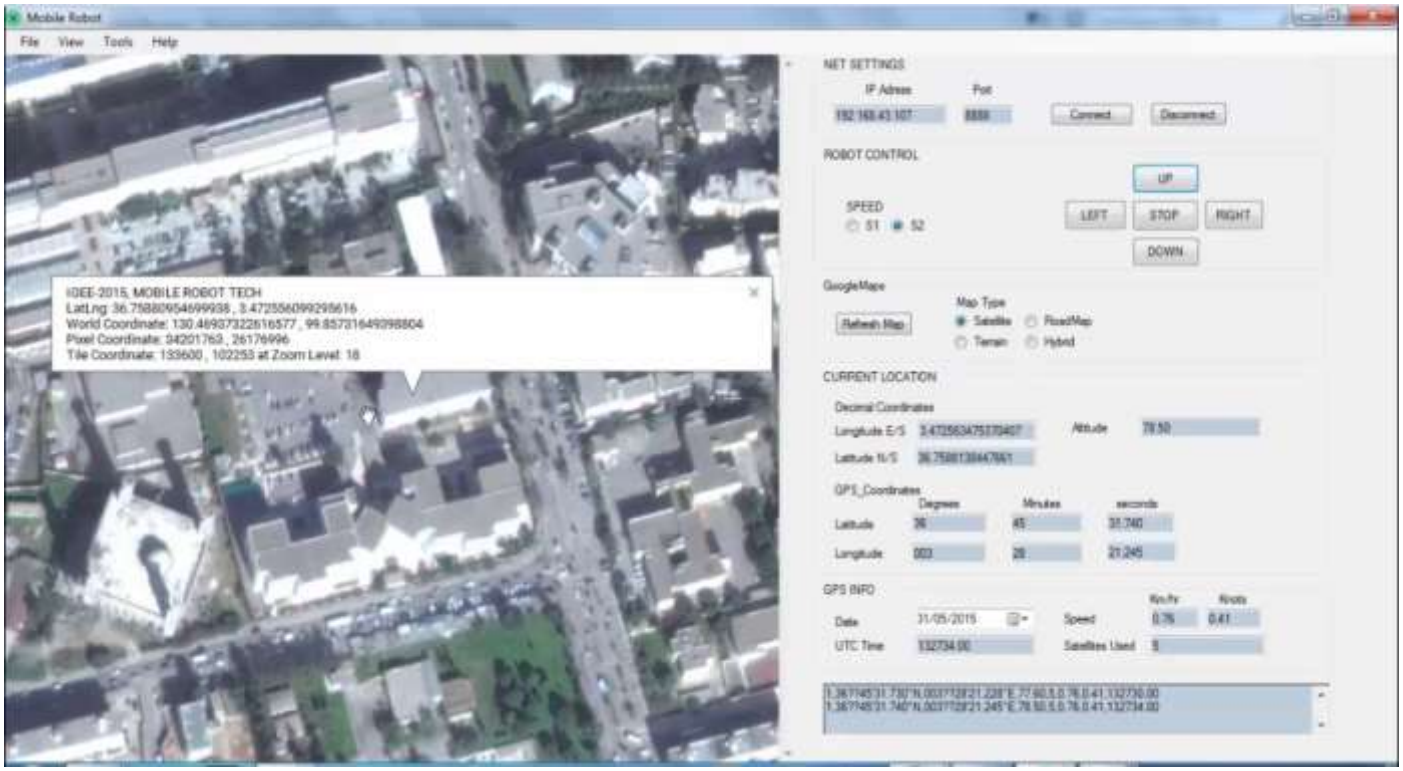


Figure 4.9 The Final GUI

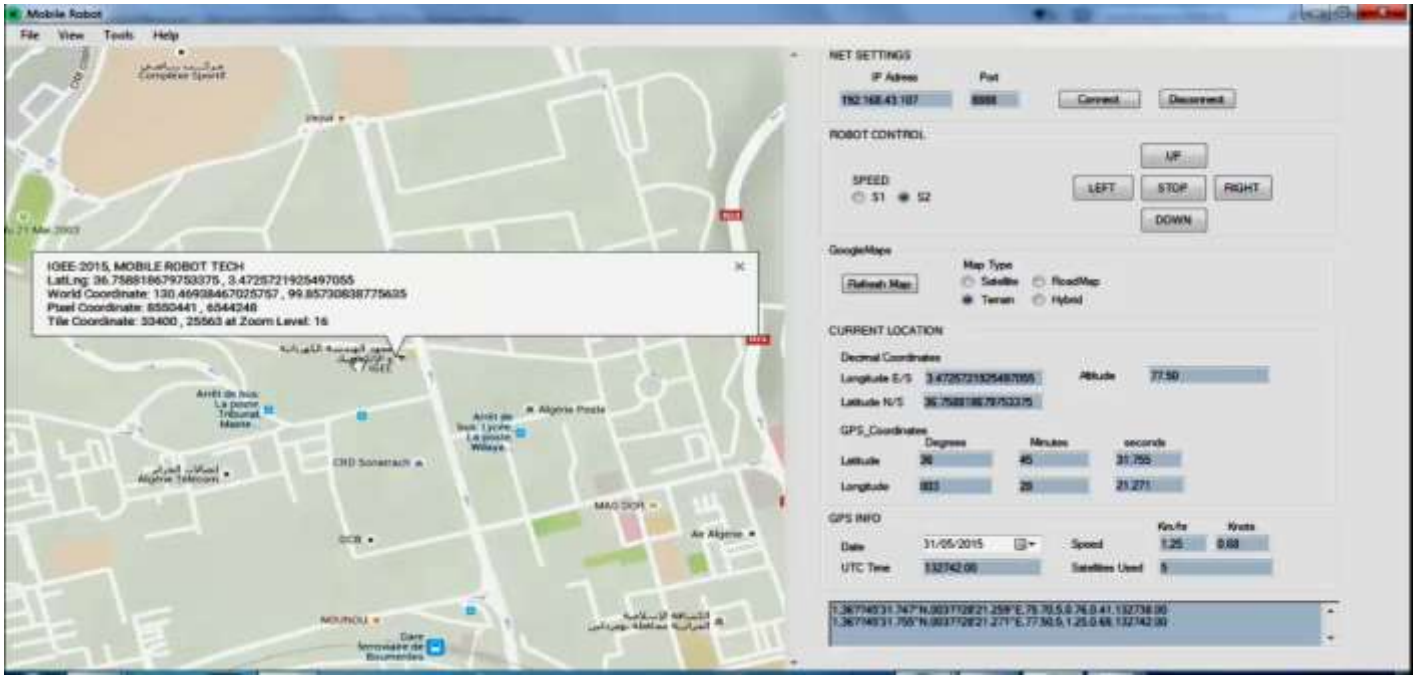


Figure 4.10 Final GUI Map View

Conclusion

The conclusion summarizes our work and exposes further development and improvements for the project.

.

Conclusion

In this project, we described the design and realization of a microcontroller based, tricycle platform Mobile Robot.

The objective of the project was to design and realize the mobile robot. The robot is designed to be controlled in real time fashion by the graphical user interface. A network is set up as the media for exchanging data and commands between the robot and user.

To achieve this work, it was necessary to:

- Choose a suitable Hardware.
- Choose the right environment to develop the graphical user interface.

The hardware design was based around the Arduino Mega as the microcontroller which is programmed using the Arduino IDE. And the graphical user interface was created in Visual C# using Visual Studio.

This project was a real investment of time and efforts, because we didn't use any preassembled parts, but we did design the platform chassis using AutoCAD in order to make it suitable for differential drive concept. The hardware design and implementation were so hard, because we were introduced to new components like GPS, and Wi-Fi Module.

In software design, we had to revise the programming language C#, and how to use visual studio in order to develop a good graphical user interface. In hardware programming, the Arduino programming language was is easy to learn because it is C-style.

In short, due to the hard work we learnt a new package of things, and gained a great experience in troubleshooting both hardware and software errors.

This project can be considered as an open project available for further developments and improvements:

- Mount an IP Camera on the mobile robot.
- Usage of IR sensors to make the robot autonomous.

- Mount a small robotic arm manipulator.
- Usage of powerful motors and batteries.
- Usage of powerful microcontroller that supports Real Time Operating Systems.

References

[1] http://www.wikipedia.com/microcontrollers

[2] http://www.ami.ac.uk/courses/ami4655_micros/u01/micro01hist.aspx.

[3] http://www.atmel.com/devices/atmega2560.aspx

[4] http://www.arduino.cc/.

[5] http://en.wikipedia.org/wiki/Arduino.

[6] http://en.wikipedia.org/wiki/Global_Positioning_System

[7] http://www.mio.com/technology-history-of-gps.htm

[8] http://techterms.com/definition/wi-fi

[9] http://compnetworking.about.com/od/networkprogramming/g/what-is-a-socket.

Appendices

a. Chassis Design