



# **EAST WEST UNIVERSITY**

**Department of the Electronics and Communication Engineering**

**Aftabnagar, Dhaka, Bangladesh.**

**December, 2015**

**Title of the Project:**

**Design and Development of an Arduino Based WiFi Robot**

**Submitted by:**

**Abdul Goni Mehedi (2012-1-55-034)**

**Zafrin Malek Mithila (2012-1-55-044)**

**Supervised by:**

**Dr. Md. Habibur Rahman**

Professor, Department of Electronics and Electrical Engineering, University of Dhaka

The project report presented in partial fulfillment of the requirement of the degree of Bachelor of Science (Engineering) in Electronics and Telecommunication Engineering.

# Declaration

---

We hereby declare that this project is based on the results found by ourselves. We have completed the project on the topic entitled “**Design and Development of Arduino Based WiFi Robot**” as well as prepared as research report to the Department of Electronic and Communication Engineering of East West University in partial fulfillment of the requirement for the degree of B.Sc. in Electronic and Telecommunication Engineering, under the supervision of Dr. Md. Habibur Rahman, Professor, Department of Electronics and Electrical Engineering, University of Dhaka.

Signature

.....  
Dr. Md. Habibur Rahman  
Professor, EEE  
University of Dhaka

Signature

.....  
Abdul Goni Mehedi  
ID: 2012-1-55-034

Signature

.....  
Zafrin Malek Mithila  
ID: 2012-1-55-044

# Acceptance

---

This project has been prepared and submitted by Abdul Goni Mehedi, ID.2012-1-55-034 and Zafrin Malek Mithila, ID.2012-1-55-044. This project report presented to the Department of Electronics and Communication Engineering, East West University is submitted in partial fulfillment of the requirement for the degree of B.Sc. in Electronics and Telecommunication Engineering, under complete supervision of the undersigned.

Signature

.....

Dr. Md. Habibur Rahman  
Professor, EEE  
University of Dhaka

# Acknowledgement

---

First of all we wish to convey our heart full thanks to **Almighty Allah** because we completed this project successfully. This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if we did not have a support of many individuals and organizations. Therefore we would like to extend our sincere gratitude to all of them. First of all we are thankful to our supervisor Dr. Md. Habibur Rahman for his logistical support and for providing necessary guidance concerning projects implementation. Without his superior knowledge and experience, the project would like in quality of outcomes, and thus his support had been essential. We would like to express our sincere thanks towards volunteer researchers on online and our senior researchers who devoted their time and knowledge in the implementation of this project.

Nevertheless, we express our gratitude toward our families and project partners for their kind co-operation and encouragement which help us in completion of this project.

# Abstract

---

A WiFi enabled remotely controllable robot is very handy in many ways. For example anyone can use such a robot as a home surveillance system remotely by only using a smartphone. In this project, a robot has been designed and developed that can be controlled by WiFi from remote place. To receive WiFi signal a WiFi shield has been interfaced with Arduino UNO microcontroller board. A motor driver has been used to change the direction of rotation and to drive the wheels of the robot. A program has been developed for the system using C+ programming language. The project has been constructed successfully and tested. The robot can be used for remote vehicle control or vehicle automation system.

## **Content:**

<b>Topics</b>	<b>Page No.</b>
<b>Chapter One:</b>	
<b>INTRODUCTION</b>	<b>8-9</b>
1.1 Motivation	9
1.2 Aim of the Project	9
1.3 Project in Brief	9
1.3.i Development of Circuit	9
1.3.ii Documentation of Data	9
<b>Chapter Two:</b>	
<b>THEORETICAL FRAMEWORK</b>	<b>10-25</b>
2.1 Microcontroller	11
2.1.i What is Microcontroller	11
2.1.ii How We Select a Microcontroller?	11
2.2 Arduino UNO	11
2.2.i Device Overview	11
2.2.ii Technical Specs	12-13
2.3 ATmega168/328-Arduino Pin Mapping	14
2.4 Arduino IDE	15
2.5 Arduino Yun	16-17
2.5.i Technical Specs	18
2.5.ii Specialized Functions	19-21
2.6 L298N Dual H-Bridge Motor Controller	22
2.6.i Specifications	23-25
2.6.ii Controls	25

## **Chapter Three:**

### **DESIGN & IMPLEMENTATION 26-39**

3.1 System Blockdiagram	27
3.2 Design of Individual Block	28-30
3.3 Completed System	31
3.4.i Software Design	32-35
3.4.ii Flow Chart of the Program	36
3.4.iii Working Procedure of the System	37-38
3.5 System Connection	39

## **Chapter Four:**

### **RESULTS 40-44**

4.1.i Graphical Representation	41
4.2.ii Observation	42-44

## **Chapter Five:**

### **Conclusion & Further Research 45-46**

5.1 Conclusion	46
5.2 Future Work	46
5.3 References	46

### **List of Figures 47**

# **Chapter One**

## **INTRODUCTION**



## **1.1 Motivation**

This project uses an Arduino to control a robot tank through a web page. The tank is driven from a control panel on a web page that is communicated to over a WiFi network connected to the internet.

## **1.2 Aim of the Project**

Every project must have a definite aim in its way. An aimless way is just like a boat without a rudder. That means the boat without rudder cannot reach ashore. The aim of the project is to have our own remote mobile home surveillance system which can be accessed from anywhere, anytime.

## **1.3 Project in Brief**

### **1.3.i Development of Circuit**

At first we have known about all the components, related to the work. Then we have gained the basic knowledge. Finally we have applied our merit and creative technical knowledge to develop the devices and the system.

### **1.3.ii Documentation of Data**

Here, all data those had been collected which have recorded and stored. We have maintained sincerity during taking and storing the data. After approval from the supervisor, all the data have been implemented into the final project report.

# **Chapter Two**

## **THEORETICAL FRAMEWORK**

## **2.1 Microcontroller**

### **2.1.i What is Microcontroller**

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. A microcontroller is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. Most programmable microcontrollers that are used today are embedded in other consumer products or machinery including phones, peripherals, automobiles and household appliances for computer systems. Due to that, another name for a microcontroller is "embedded controller." Some embedded systems are more sophisticated, while others have minimal requirements for memory and programming length and a low software complexity. Input and output devices include solenoids, LCD displays, relays, switches and sensors for data like humidity, temperature or light level, amongst others.

### **2.1.ii How we select a Microcontroller?**

- ✓ Amount of RAM and ROM.
- ✓ Reliable sources of MCU.
- ✓ Cost effective.
- ✓ Maximum speed of microcontroller.
- ✓ Availability of software and hardware.
- ✓ A timer module that allow working on real time.
- ✓ A serial I/O port.
- ✓ An ADC to allow acceptance of analog input data for processing.

## **2.2 Arduino UNO**

### **2.2.i Device Overview**

The UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.



Fig 2.1: Arduino UNO.

"UNO" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The UNO board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The UNO board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform.

Arduino is a tool for making computers that can sense and control more of the physical world than our desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

## 2.2.ii Technical Specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g



## 2.3 ATmega168/328-Arduino Pin Mapping

Note that this chart is for the DIP-package chip. The Arduino Mini is based upon a smaller physical IC package that includes two extra ADC pins, which are not available in the DIP-package Arduino implementations.

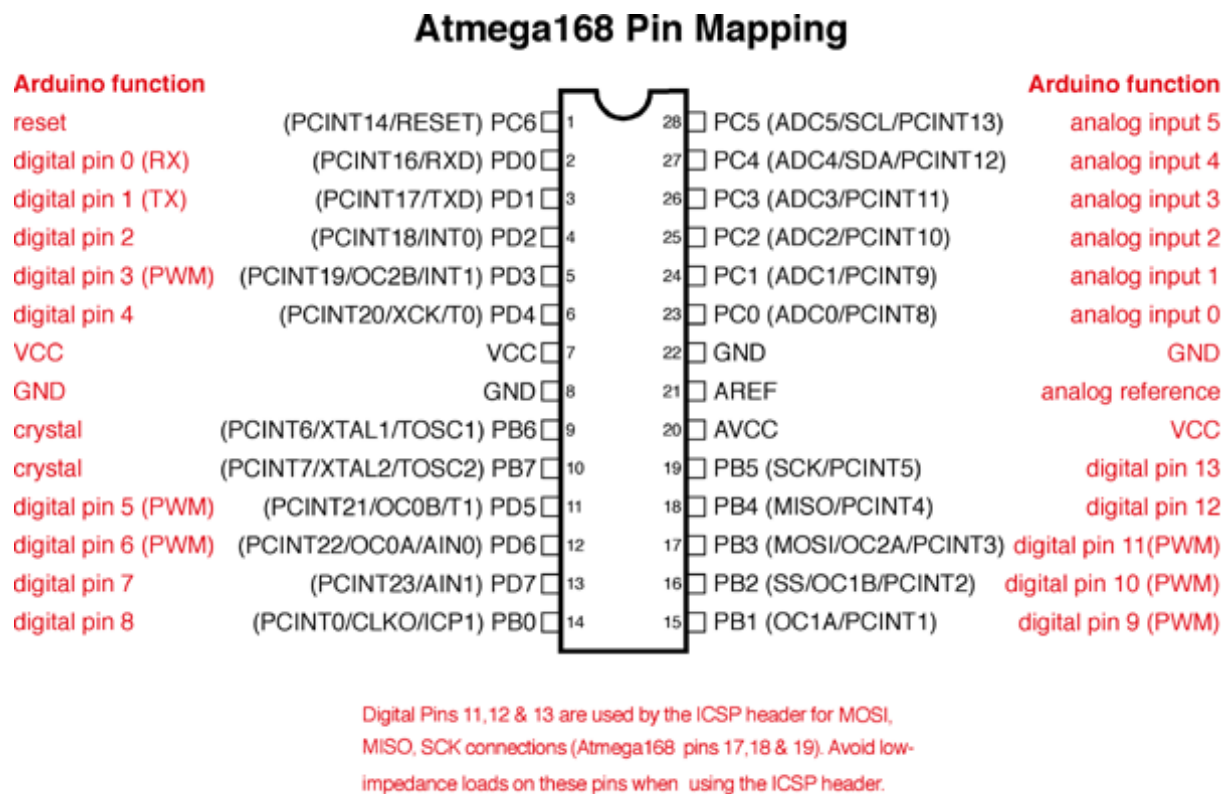


Fig 2.3: Arduino Pin Mapping.

## 2.4 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

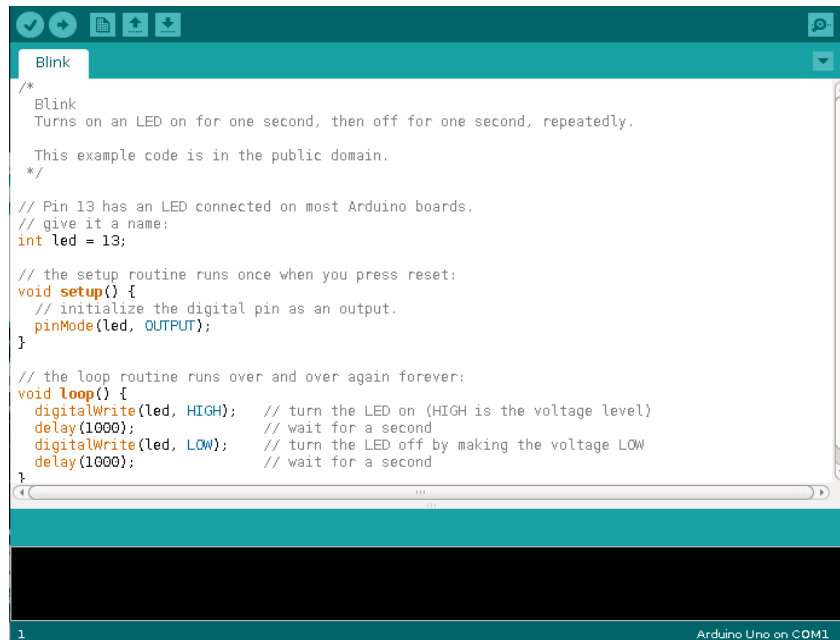


Fig 2.4: Arduino IDE Window.

1. Upload: Compiles our code and uploads it to the configured board.
2. Verify: Checks our code for errors compiling it.
3. New: Creates a new sketch.
4. Open: Presents a menu of all the sketches in our sketchbook. Clicking one will open it within the current window overwriting its content.
5. Save: Saves our sketch.
6. Serial Monitor: Opens the serial monitor.

## 2.5 Arduino Yun

The Arduino Yún is a micro-controller board based on the ATmega32u4 and the Atheros AR9331. The Atheros processor supports a Linux distribution based on OpenWrt named OpenWrt-Yun. The board has built-in Ethernet and WiFi support, a USB-A port, micro-SD card slot, 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a 3 reset buttons.

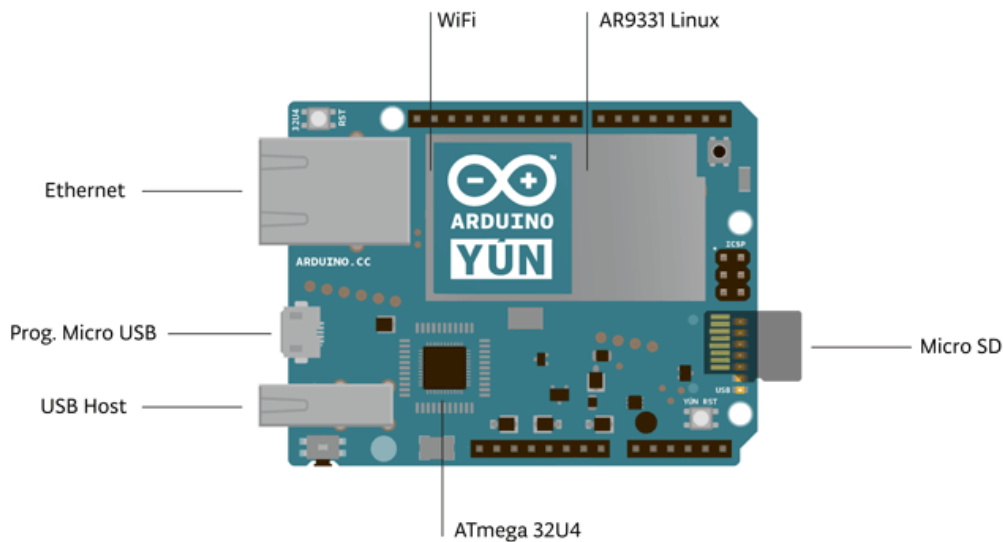


Fig 2.5: Arduino YUN Shield.

**NB:- In some countries, it is prohibited to sell WiFi enabled devices without government approval. While waiting for proper certification, some local distributors are disabling WiFi functionality. Check with your dealer before purchasing a Yún if you believe you may live in such a country. If you wish to disable WiFi, run this sketch. For more information, refer to this forum post.**

The Yún distinguishes itself from other Arduino boards in that it can communicate with the Linux distribution onboard, offering a powerful networked computer with the ease of Arduino. In addition to Linux commands like CURL, you can write your own shell and python scripts for robust interactions.



The Yún is similar to the Leonardo in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Yún to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port.

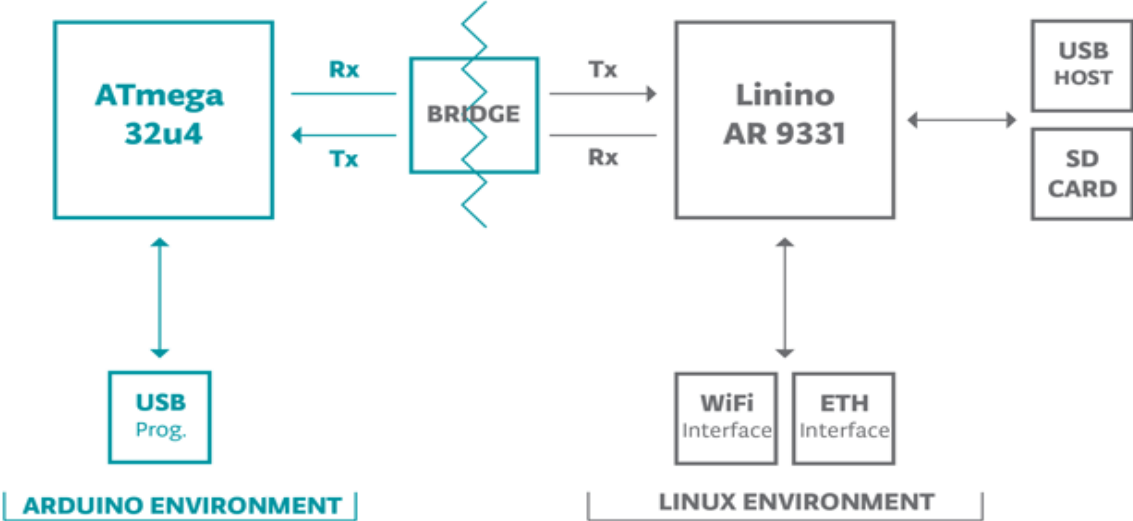


Fig 2.6: YUN arduino and linux environment.

The Bridge library facilitates communication between the two processors, giving Arduino sketches the ability to run shell scripts, communicate with network interfaces, and receive information from the AR9331 processor. The USB host, network interfaces and SD card are not connected to the 32U4, but the AR9331, and the Bridge library also enables the Arduino to interface with those peripherals [3].

## 2.5.i Technical Specs

Because the Yun has two processors, this section shows the characteristics of each one in two separate tables.

### AVR Arduino Microcontroller

Microcontroller	ATmega32U4
Operating Voltage	5V
Input Voltage	5
Digital I/O Pins	20
PWM Channels	7
Analog Input Pins	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 4 KB used by bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz

### Linux Microprocessor

Processor	Atheros AR9331
Architecture	MIPS @400MHz
Operating Voltage	3.3V
Ethernet	IEEE 802.3 10/100Mbit/s
WiFi	IEEE 802.11b/g/n
USB Type-A	2.0 Host
Card Reader	Micro-SD only
RAM	64 MB DDR2
Flash Memory	16 MB
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz
PoE compatible 802.3af card support	See <i>Power</i>

## 2.5.ii Specialized Functions

- **Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data using the ATmega32U4 hardware serial capability. Note that on the Yún, the Serial class refers to USB (CDC) communication; for TTL serial on pins 0 and 1, use the Serial1 class. The hardware serials of the ATmega32U4 and the AR9331 on the Yún are connected together and are used to communicate between the two processors. As is common in Linux systems, on the serial port of the AR9331 is exposed the console for access to the system, this means that you can access to the programs and tools offered by Linux from your sketch.
- **TWI:** 2 (SDA) and 3 (SCL). Support TWI communication using the Wire library.
- **External Interrupts:** 3 (interrupt 0), 2 (interrupt 1), 0 (interrupt 2), 1 (interrupt 3) and 7 (interrupt 4). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details. Is not recommended to use pins 0 and 1 as interrupts because they are the also the hardware serial port used to talk with the Linux processor. Pin 7 is connected to the AR9331 processor and it may be used as handshake signal in future. Is recommended to be careful of possible conflicts if you intend to use it as interrupt.
- **PWM:** 3, 5, 6, 9, 10, 11, and 13. Provide 8-bit PWM output with the analogWrite() function.
- **SPI:** on the ICSP header. These pins support SPI communication using the SPI library. Note that the SPI pins are not connected to any of the digital I/O pins as they are on the UNO, They are only available on the ICSP connector. This means that if you have a shield that uses SPI, but does NOT have a 6-pin ICSP connector that connects to the Yún's 6-pin ICSP header, the shield will not work. The SPI pins are also connected to the AR9331 gpio pins, where it has been implemented in software the SPI interface. This means that the ATmega32u4 and the AR9331 can also communicate using the SPI protocol.

- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- There are several other status LEDs on the Yún, indicating power, WLAN connection, WAN connection and USB.

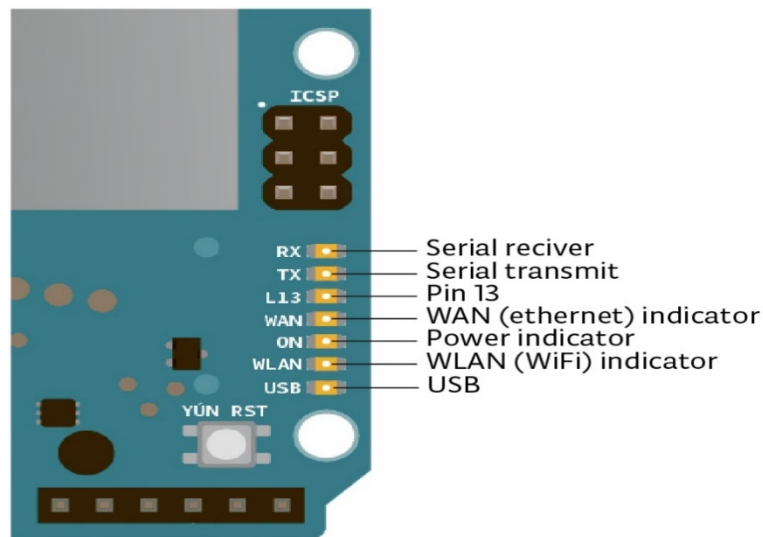


Fig 2.7: YUN Indicator LED

- Analog Inputs: A0 - A5, A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12). The Yún has 12 analog inputs, labeled A0 through A11, all of which can also be used as digital i/o. Pins A0-A5 appear in the same locations as on the UNO; inputs A6-A11 are on digital i/o pins 4, 6, 8, 9, 10, and 12 respectively. Each analog input provide 10 bits of resolution (i.e. 1024 different values). By default the analog inputs measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.
- AREF. Reference voltage for the analog inputs. Used with analogReference().

There are 3 reset buttons with different functions on the board:

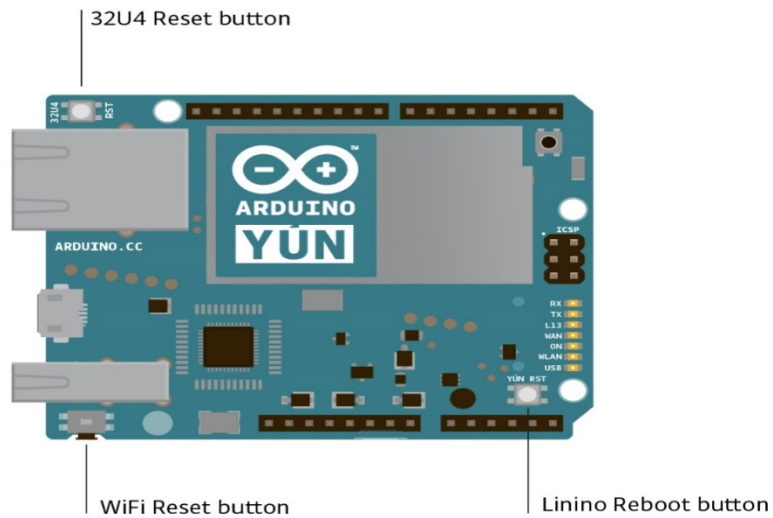


Fig 2.8: Yun Reset buttons.

- **Yún RST.** Bring this line LOW to reset the AR9331 microprocessor. Resetting the AR9331 will cause the reboot of the linux system. All the data stored in RAM will be lost and all the programs that are running will be terminated.
- **32U4 RST.** Bring this line LOW to reset the ATmega32U4 microcontroller. Typically used to add a reset button to shields which block the one on the board.
- **WLAN RST.** This button has a double feature. Primarily serves to restore the WiFi to the factory configuration. The factory configuration consist to put the WiFi of the Yún in access point mode (AP) and assign to it the default IP address that is 192.168.240.1, in this condition you can connect with your computer to the a WiFi network that appear with the SSID name "Arduino Yun-XXXXXXXXXXXX", where the twelve 'X' are the MAC address of your Yún. Once connected you can reach the web panel of the Yún with a browser at the 192.168.240.1 or "http://arduino.local" address. Note that restoring the WiFi configuration will cause the reboot of the linux environment. To restore your WiFi configuration you have to press and hold the WLAN RST button for 5 seconds. When you press the button the WLAN blue LED will start to blink and will keep still blinking when you release the button after 5 seconds indicating that the WiFi restore procedure has been recorded. The second function of the WLAN RST button is to restore the linux image to the default factory image. To restore the linux environment you must press the button for 30 seconds. Note that restoring the factory image make you lose all the files saved and softwares installed on the on-board flash memory connected to the AR9331.

## 2.6 L298N Dual H-Bridge Motor Controller

H-Bridge's are typically used in controlling motors speed and direction, but can be used for other projects such as driving the brightness of certain lighting projects such as high powered LED arrays.

### How it works:

An H-Bridge is a circuit that can drive a current in either polarity and be controlled by \*Pulse Width Modulation (PWM).

\* Pulse Width Modulation is a means in controlling the duration of an electronic pulse. In motors try to imagine the brush as a water wheel and electrons as a the flowing droplets of water. The voltage would be the water flowing over the wheel at a constant rate, the more water flowing the higher the voltage. Motors are rated at certain voltages and can be damaged if the voltage is applied to heavily or if it is dropped quickly to slow the motor down. Thus PWM. Take the water wheel analogy and think of the water hitting it in pulses but at a constant flow. The longer the pulses the faster the wheel will turn, the shorter the pulses, the slower the water wheel will turn. Motors will last much longer and be more reliable if controlled through PWM.

### Pins:

- Out 1: Motor A lead out
- Out 2: Motor A lead out
- Out 3: Motor B lead out
- Out 4: Mo (*Can actually be from 5v-35v, just marked as 12v*)
- GND: Ground
- 5v: 5v input (*unnecessary if your power source is 7v-35v, if the power source is 7v-35v then it can act as a 5v out*)
- EnA: Enables PWM signal for Motor A (Please see the "Arduino Sketch Considerations" section)
- In1: Enable Motor A
- In2: Enable Motor A
- In3: Enable Motor B
- In4: Enable Motor B
- EnB: Enables PWM signal for Motor B (Please see the "Arduino Sketch Considerations" section)

## 2.6.i Specifications

- Double H bridge Drive Chip: *L298N*
- Logical voltage: *5V Drive voltage: 5V-35V*
- Logical current: *0-36mA Drive current: 2A (MAX single bridge)*
- Max power: *25W*
- Dimensions: *43 x 43 x 26mm*
- Weight: *26g*

*\*Built-in 5v power supply, when the driving voltage is 7v-35v*

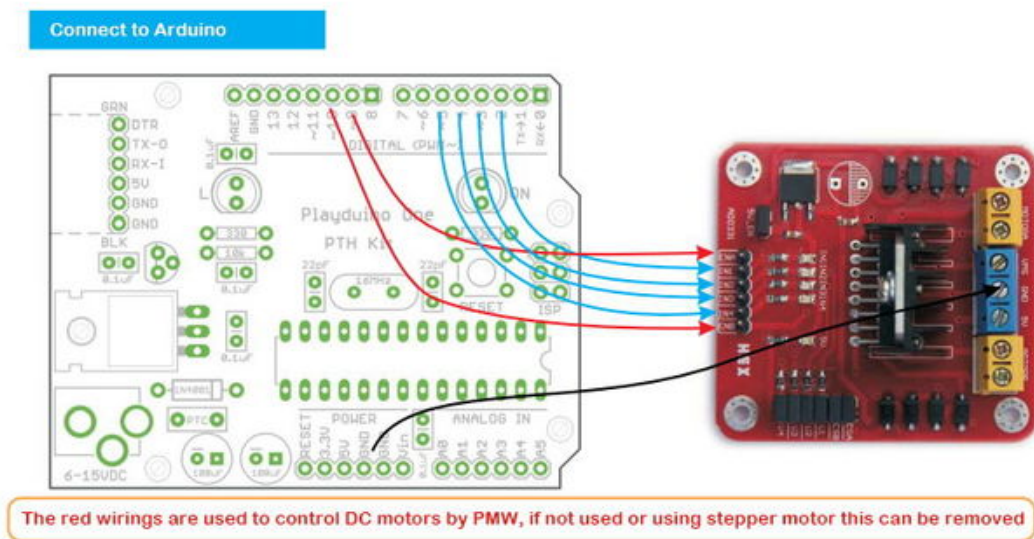


Fig 2.9: Motor Drive PWM.

There are several different models of these L298N Dual H-Bridge Motor Controllers. The generic wiring schematic above should do the trick for most [4].

### Two things to mention:

- Make sure you have all of our grounds tied together; Arduino, Power source, and the Motor controller.
- The PWM Pins are unnecessary if we do not want to control PWM features.

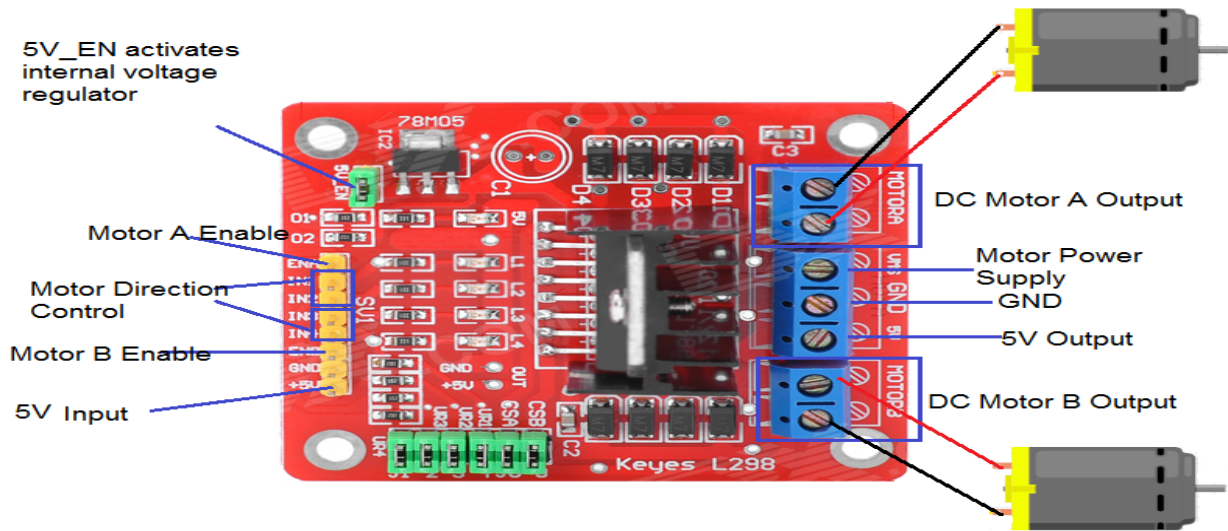


Fig 2.10: L298N Motor drive connection.

### Arduino Sketch Considerations:

The Arduino code sketch is pretty straight forward. Since there isn't a library for the L298N Dual H-Bridge Motor Controller we just have to declare which pins the controller is hooked to.

The “**intdir (number) Pin(letter)**” pins can be connected to any available digital pin you have available, as long as you declare the correct pin in your sketch. This makes the L298N Dual H-Bridge Motor Controller very versatile if your project is using a lot of Arduino pins.

The **int “speed Pin (letter)”** pins need to be connected to a PWM pin on the Arduino if we want to enable speed control through PWM.

As a quick cheat we have included a list of PWM pins for the main two types of Arduino's we use:

- **AT MEGA – PWM:** 2 to 13 and 44 to 46. Provide 8-bit PWM output with the `analogWrite()` function.
- **UNO – PWM:** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the `analogWrite()` function.



## Arduino Sketch Example:

This code example we wrote to allow a serial monitor program such as Putty to control the L298N Dual H-Bridge Motor Controller via a keyboard with key presses.

## 2.6.ii Controls

### Key ..... Motor

- 1 ..... Motor 1 Forward
- 2 ..... Motor 1 Stop
- 3 ..... Motor 1 Reverse
- 4 ..... Motor 2 Forward
- 5 ..... Motor 2 Stop
- 6 ..... Motor 2 Reverse

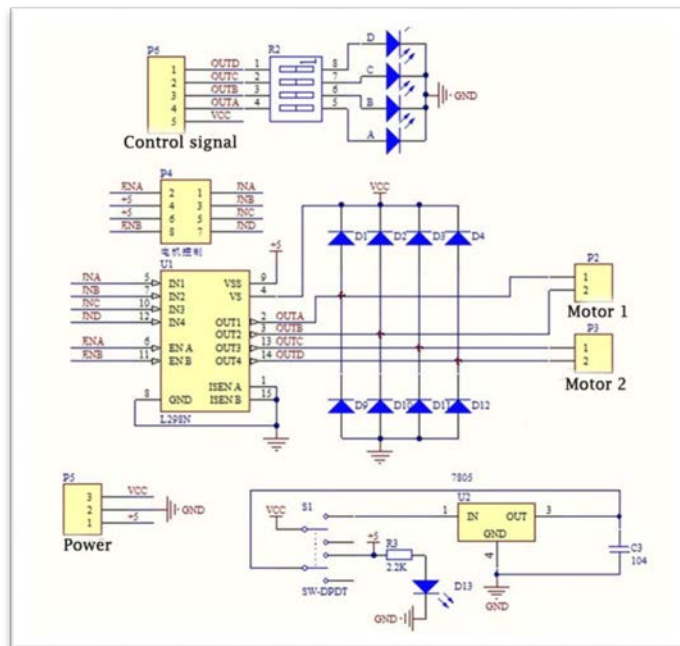


Fig 2.11: Motor drive Circuit.

# **Chapter Three**

## **DESIGN & IMPLEMENTATION**

## 3.1 System Blockdiagram

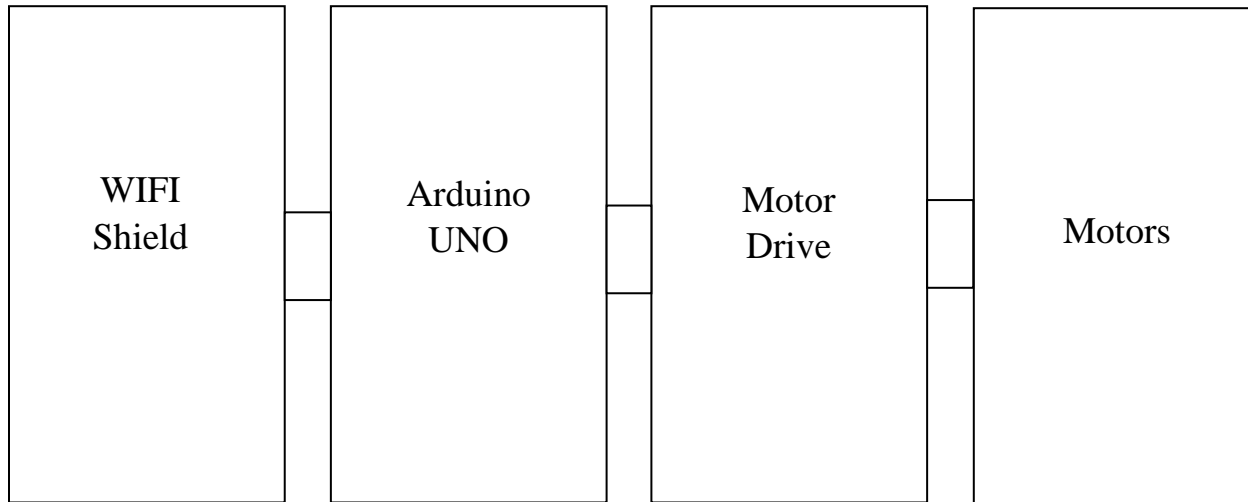


Fig 3.1: System block diagram.

Here in the system there are mainly four psrts that workes together.

1. WiFi Shield
2. Arduino UNO
3. Motor Drive
4. Motors

The WiFi shield lisitens to the commands given to the web server through serial monitor. Arduino UNO holds the codes for controling the motors. When a command is received the UNO outputs the signal according to the given code to it to execute. Although UNO cant control the way of rotation of motors, so a motor drive is used to simplify the motor controls. It acts as a H bridge motor control for two motors.

## 3.2 Design of Individual Block

### Arduino UNO and YUN Shield

Arduino UNO and YUN Shield are stacked on each other. The UNO holds all the basic codes to run the Dual H Bridge Motor Drive. On the other hand YUN establishes and maintains network connection through its WiFi module which will be connected to internet through a router. Thus any remote device can access the Arduino and control it.

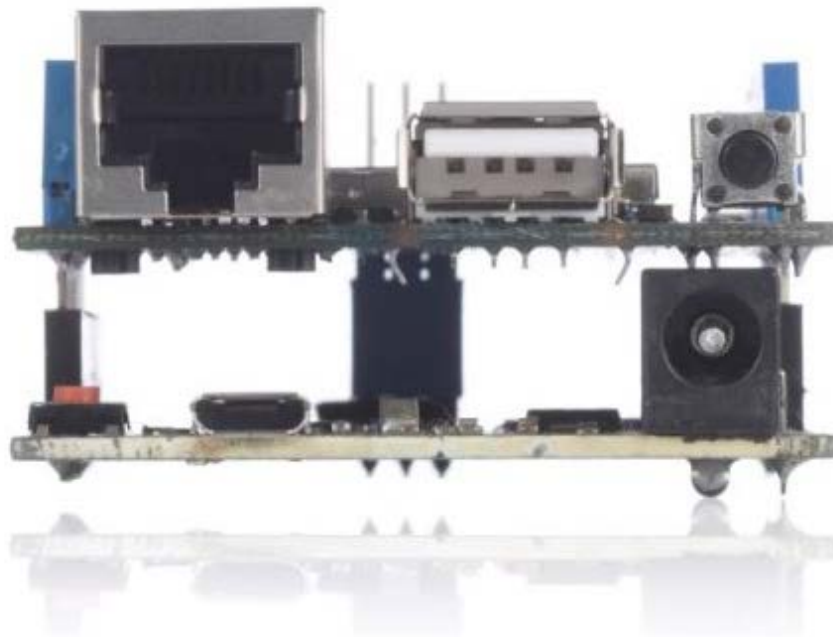


Fig 3.2: Arduino UNO and Yun Connection.

In order to function properly the Arduino UNO must be connected to a DC source of constant 5V. It is also mandatory to connect the antenna of Arduino YUN in order to get more network area coverage.

## L298N Dual H Bridge Motor Drive

The L298N Dual H Bridge Motor Drive is a shield used to drive two motors at any directions with variable speed functions. As in this project we are not using any stepper motor but only simple DC gear motor we use the PWM signal from Arduino pin 9 & 6 to enable motor 1 and motor 2. The value of PWM is set to 80 to get a decent rate of acceleration.

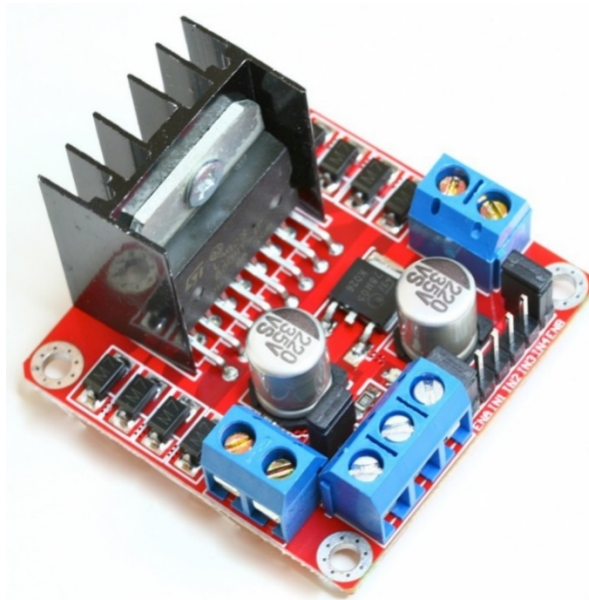
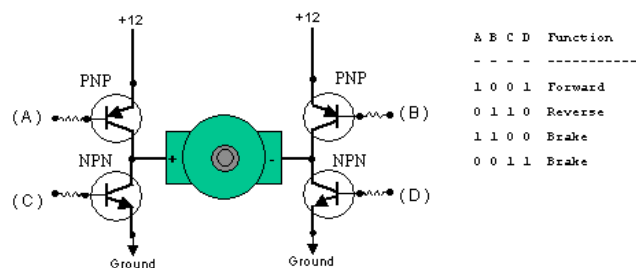


Fig 3.3: L298N Motor drive.

A DC 9V battery should be connected to the power port and the GND must be also connected to Arduino UNO's GND port. The enable pins(ENA & ENB) of both motors will be connectd to PWM pins 9 & 6 of Arduino. And the direction pins(IN1, IN2, IN3, IN4) to pin 8, 7, 5, 4 accordingly to Arduino.



## 210:1 Micro Metal Gearmotor with Chassis

The two DC motors, motor A and motor B will have one positive and one negative pole wire will be connected to the L298N Dual H Bridge Motor Drive's OUT2 and OUT3 ports.



Fig 3.4: Micro Metal Gearmotor



Fig 3.5: Chassis

### 3.3 Completed System

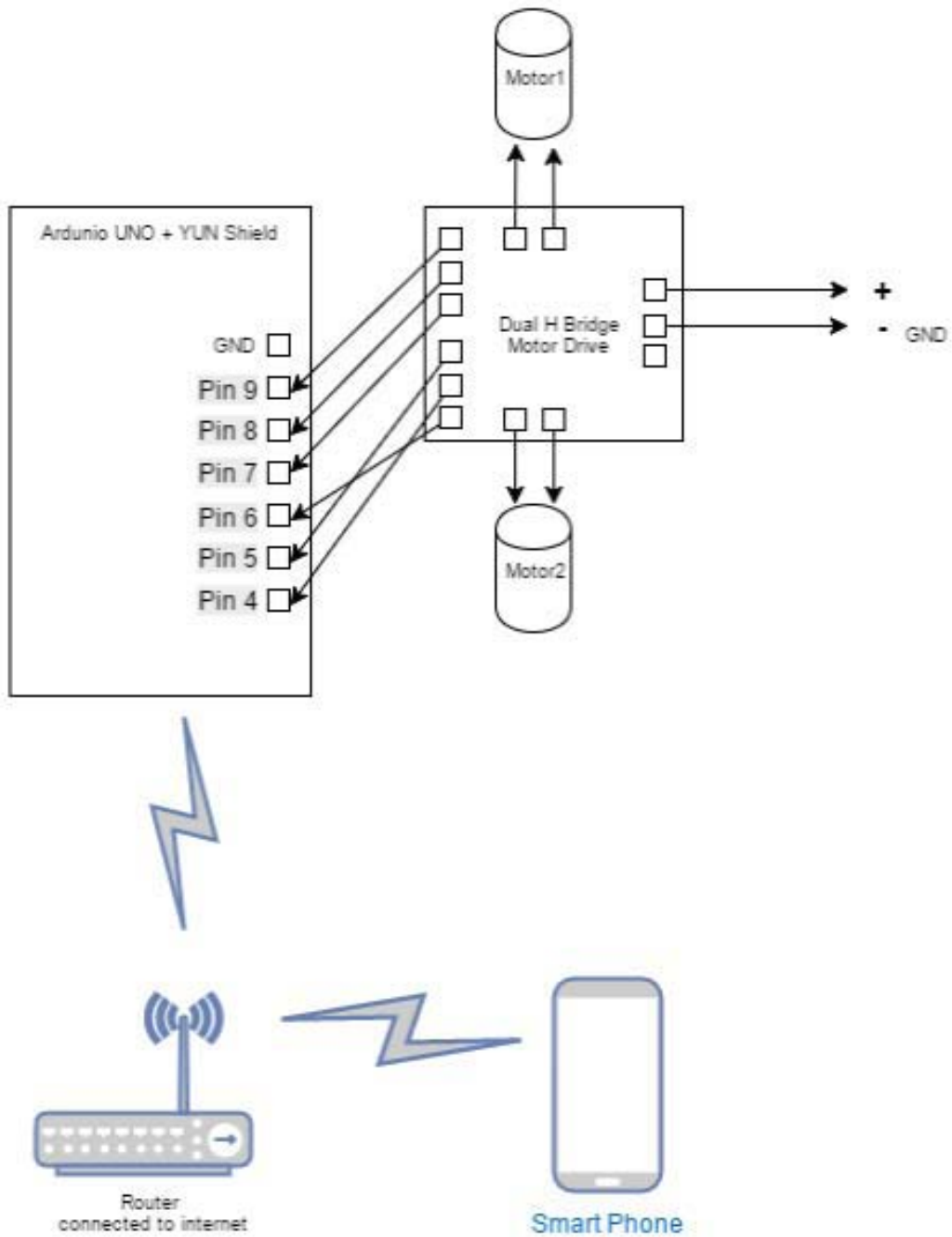
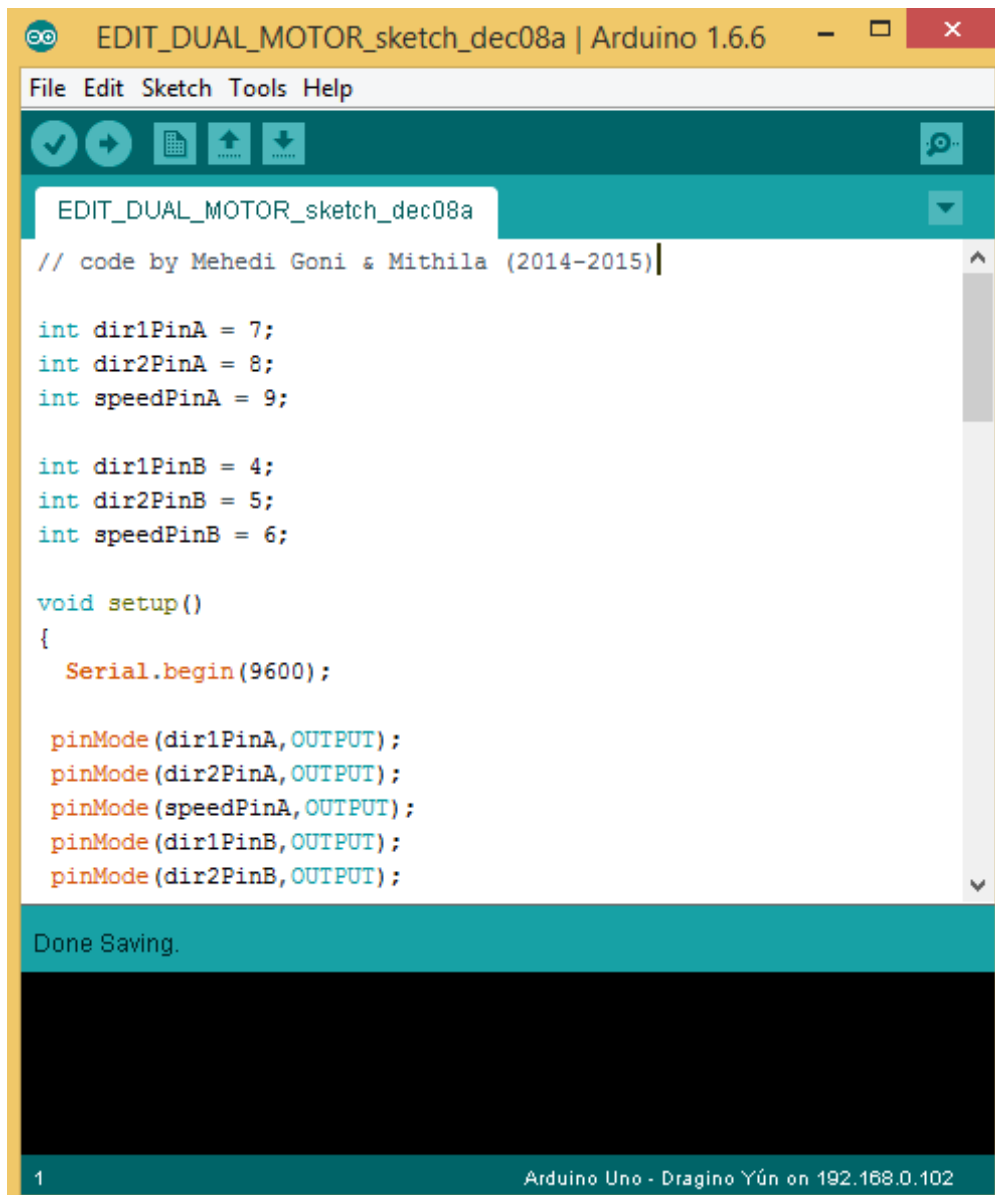


Fig 3.6: Completed system diagram.

### 3.4.i Software Design



```
EDIT_DUAL_MOTOR_sketch_dec08a | Arduino 1.6.6
File Edit Sketch Tools Help
EDIT_DUAL_MOTOR_sketch_dec08a
// code by Mehedi Goni & Mithila (2014-2015)

int dir1PinA = 7;
int dir2PinA = 8;
int speedPinA = 9;

int dir1PinB = 4;
int dir2PinB = 5;
int speedPinB = 6;

void setup()
{
  Serial.begin(9600);

  pinMode(dir1PinA, OUTPUT);
  pinMode(dir2PinA, OUTPUT);
  pinMode(speedPinA, OUTPUT);
  pinMode(dir1PinB, OUTPUT);
  pinMode(dir2PinB, OUTPUT);
}

Done Saving.

1 Arduino Uno - Dragino Yún on 192.168.0.102
```

Fig 3.7: Software design IDE.

The software is very simple. It contains two part. One part to control the motors and another part to run the web server required for remote control access.



## Software code:

// code by Abdul Goni Mehedi & Zafrin Malek Mithila (2014-2015)

**intdir1PinA = 7;**

**intdir2PinA = 8;**

**intspeedPinA = 9;**

**intdir1PinB = 4;**

**intdir2PinB = 5;**

**intspeedPinB = 6;**

**void setup()**

**{**

**Serial.begin(9600);**

**pinMode(dir1PinA,OUTPUT);**

**pinMode(dir2PinA,OUTPUT);**

**pinMode(speedPinA,OUTPUT);**

**pinMode(dir1PinB,OUTPUT);**

**pinMode(dir2PinB,OUTPUT);**

**pinMode(speedPinB,OUTPUT);**

**}**

**void loop()**

**{**

**if (Serial.available() >0)**

**{ intinByte = Serial.read();**

**int speed;**

```
switch (inByte) {

//_____Motor 1_____
case '1': //Forward M1
analogWrite(speedPinA,80);
digitalWrite(dir2PinA,HIGH);
digitalWrite(dir1PinA,LOW);
delay (1000);
break;

case '2': //Stop M1
analogWrite(speedPinA,0);
digitalWrite(dir2PinA,HIGH);
digitalWrite(dir1PinA,LOW);
delay (1000);
break;

case '3': //Reverse M1
analogWrite(speedPinA,80);
digitalWrite(dir2PinA,LOW);
digitalWrite(dir1PinA,HIGH);
delay (1000);
break;

//_____Motor 2_____
case '4': //Forward M2
analogWrite(speedPinB,80);
digitalWrite(dir2PinB,HIGH);
```

```
digitalWrite(dir1PinB,LOW);  
delay (1000);  
break;
```

```
case '5': //Stop M2  
analogWrite(speedPinB,0);  
digitalWrite(dir2PinB,HIGH);  
digitalWrite(dir1PinB,LOW);  
delay (1000);  
break;
```

```
case '6': //Reverse M2  
analogWrite(speedPinB,80);  
digitalWrite(dir2PinB,LOW);  
digitalWrite(dir1PinB,HIGH);  
delay (1000);  
break;  
}
```

```
}}  
// END
```

```
[5] [6]
```

### 3.4.ii Flow Chart

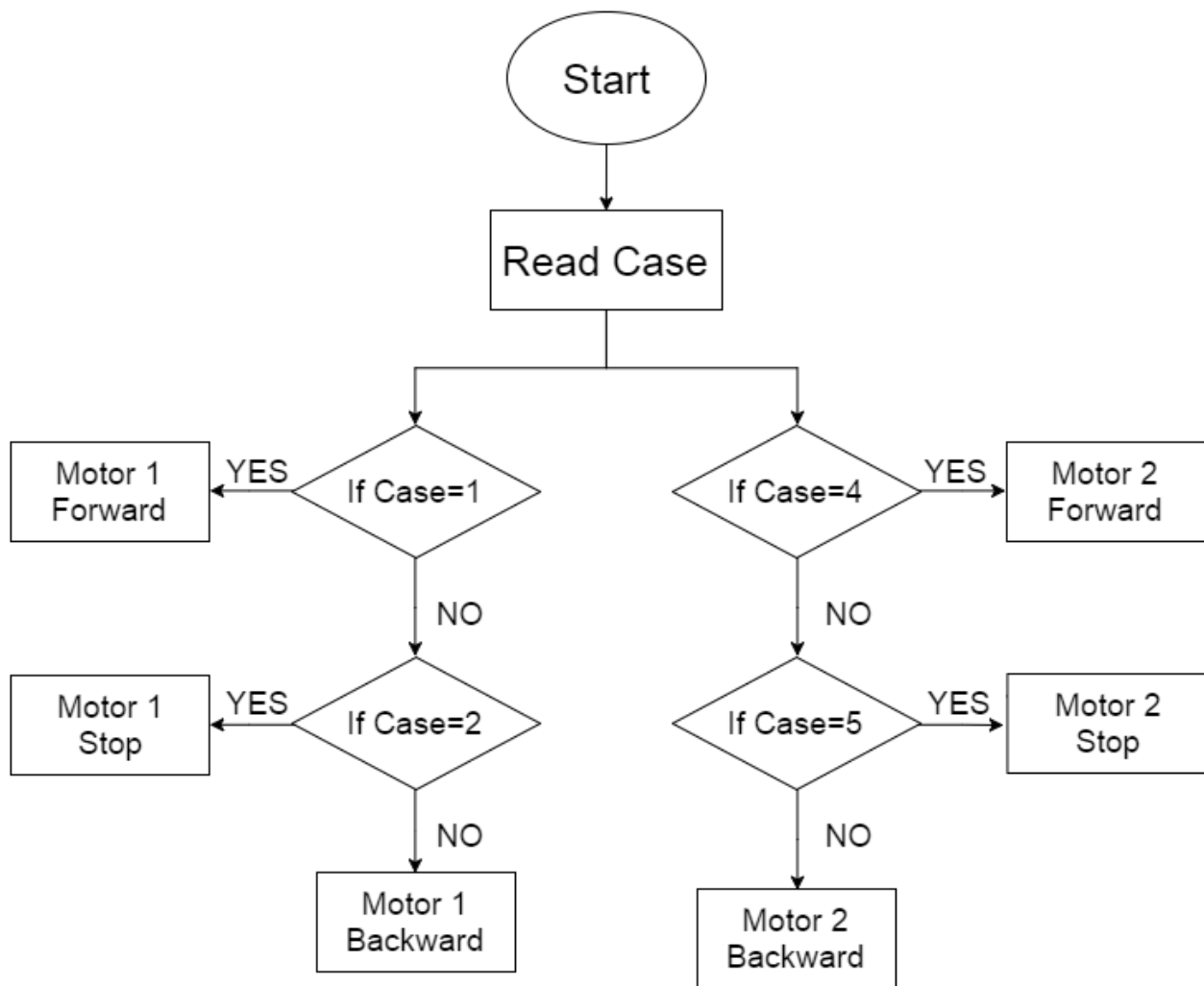



Fig 3.8: System control flow chart.

Here in this flow chart it is showing how the Arduino UNO will control the motors with the given program and commands.

### 3.4.iii Working Procedure of the System

First to observe the working procedure we upload the program into the Arduino UNO pressing upload button on Arduino IDE.



```
EDIT_DUAL_MOTOR_sketch_dec08a | Arduino 1.6.6
File Edit Sketch Tools Help
EDIT_DUAL_MOTOR_sketch_dec08a
// code by Mehedi Goni & Mithila (2014-2015)

int dir1PinA = 7;
int dir2PinA = 8;
int speedPinA = 9;

int dir1PinB = 4;
int dir2PinB = 5;
int speedPinB = 6;

void setup()
{
  Serial.begin(9600);

  pinMode(dir1PinA, OUTPUT);
  pinMode(dir2PinA, OUTPUT);
  pinMode(speedPinA, OUTPUT);
  pinMode(dir1PinB, OUTPUT);
  pinMode(dir2PinB, OUTPUT);
}

Done uploading.

Sketch uses 2,662 bytes (8%) of program storage space. Maximum is 32,
Global variables use 194 bytes (9%) of dynamic memory, leaving 1,854
1 - 12 Arduino/Genuino Uno on COM4
```

Fig 3.9: Software Upload IDE.

Then we open the serial monitor on the Arduino IDE. To execute the control commands we now only have to send those commands through the serial monitor window.

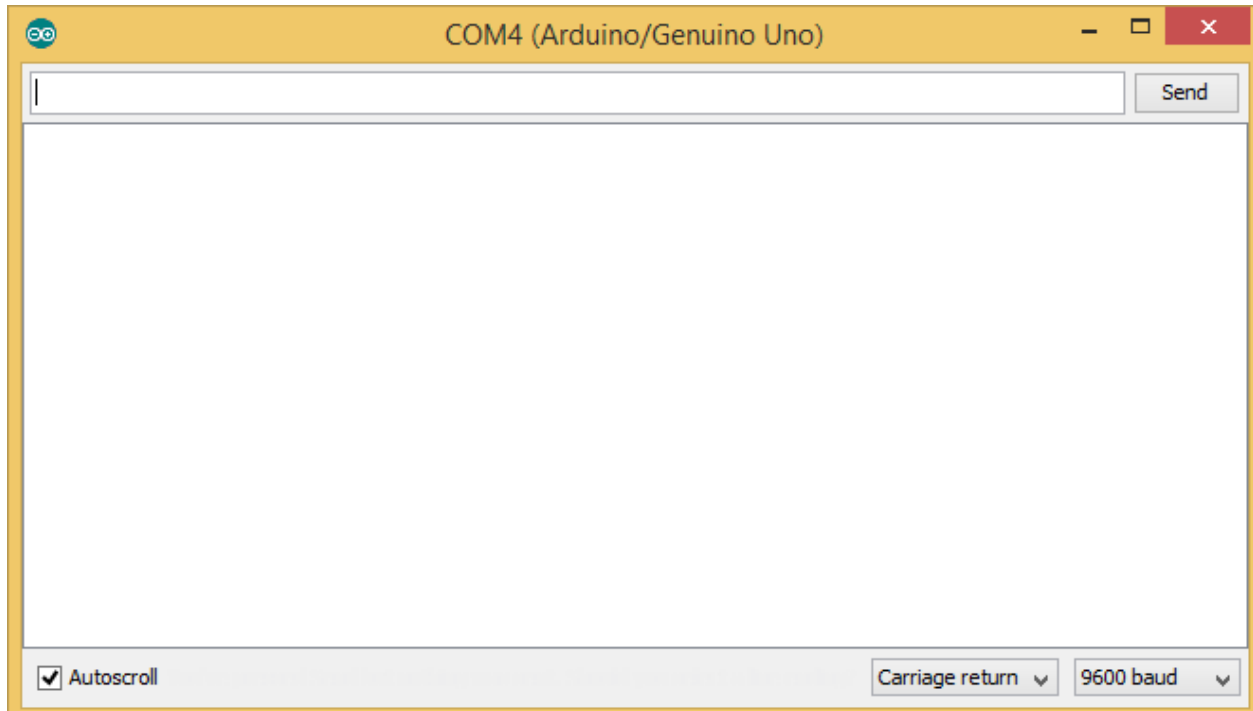


Fig 3.10: Serial Monitor.

The commands are given below:

Send 1 for Motor 1 forward

Send 2 for Motor 1 stop

Send 3 for Motor 1 backward

Send 4 for Motor 2 forward

Send 5 for Motor 2 stop

Send 6 for Motor 2 backward

### 3.5 System Connection (Scematic Diagram)

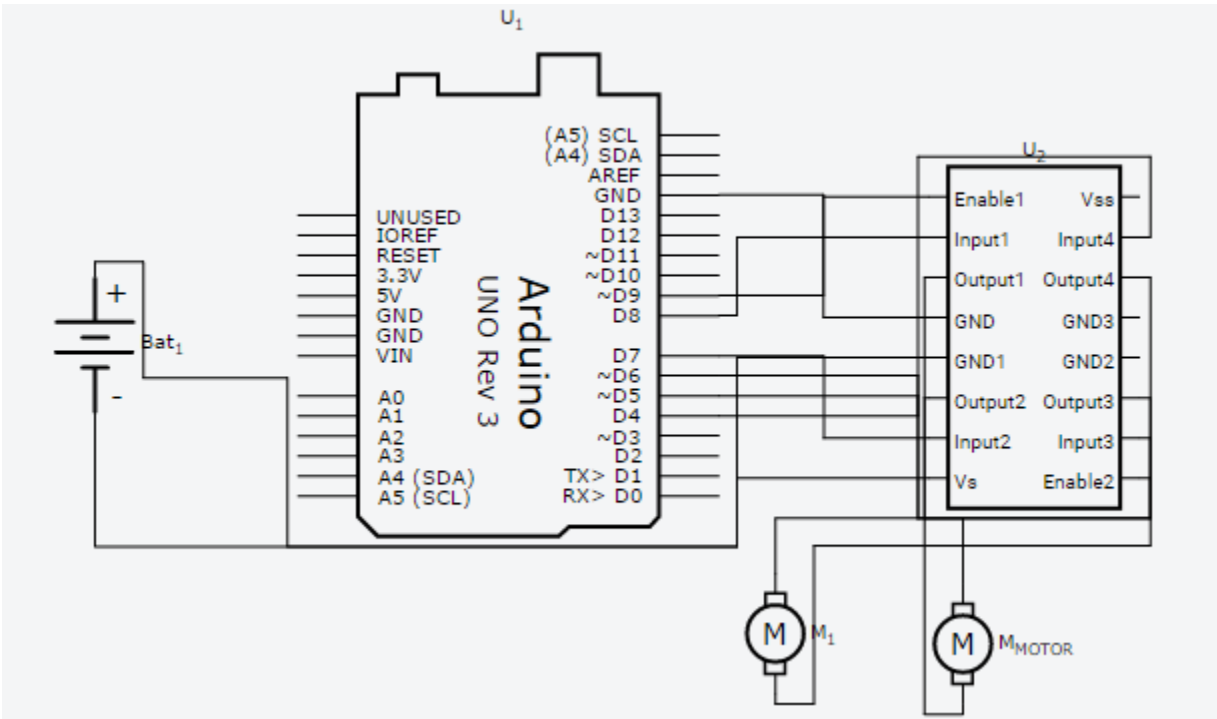


Fig 3.11: System Connection Scematic Diagram

The Arduino UNO and Yun are powered by 5V DC current source and the motor drive requires 9V dc source. Yun is connected to the internet through a WiFi network. Pins (D9, D8, D7, D6, D5, D4) of UNO are connected to the motor drive (Enable 1, Input 1, Input 2, Enable 2, Input 3, Input 4) pins. The motors are connected to the motor drive by (Output 1, Output 2, Output 3, Output 4) pins.

# **Chapter Four**

## **RESULTS**



## 4.1.i Graphical Representation

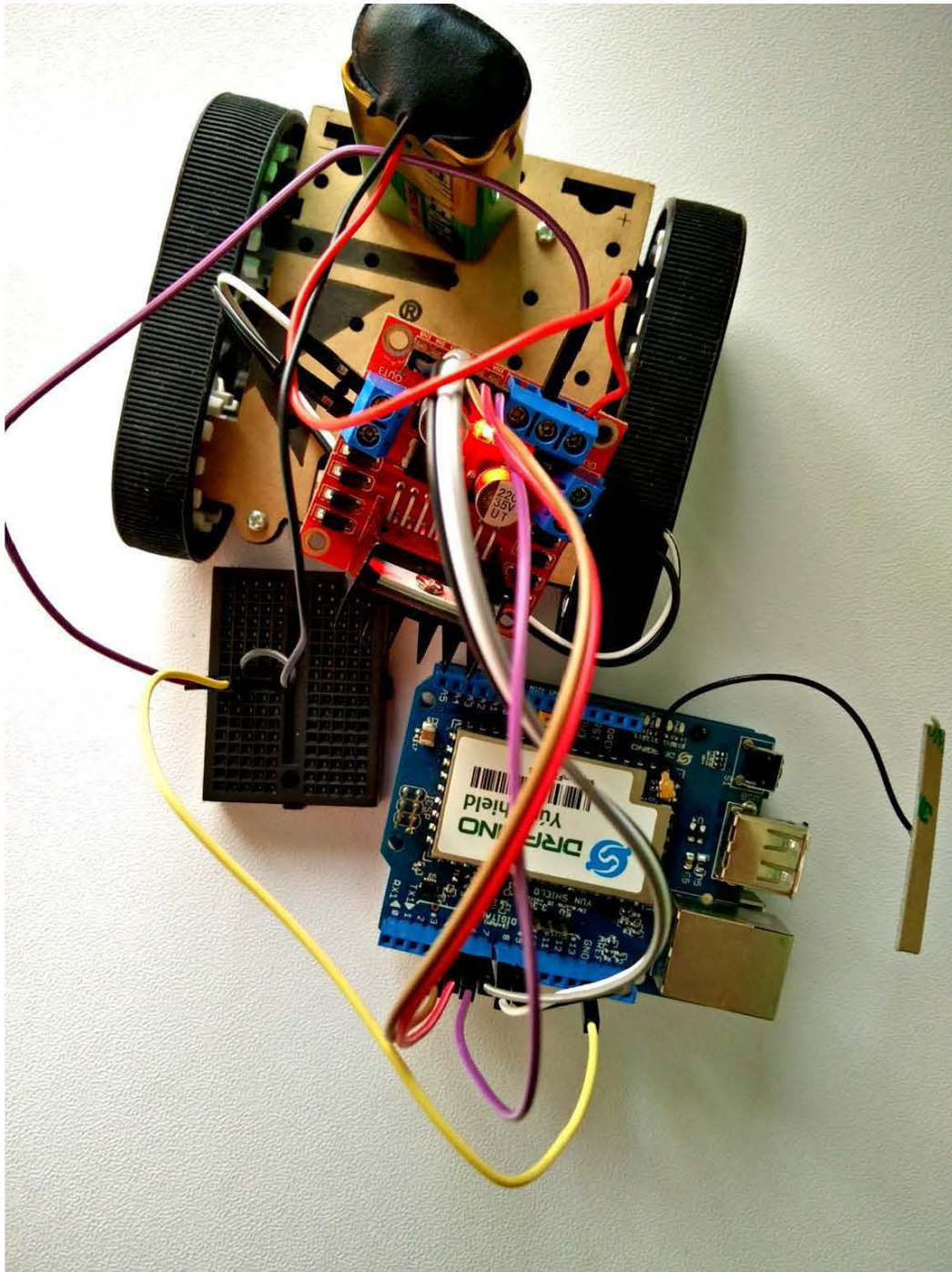


Fig 4.1: WiFi Controlled Robot (Full Project graphical representation)

## 4.2.ii Observation

Pressing 1 at serial monitor motor 1 goes forward

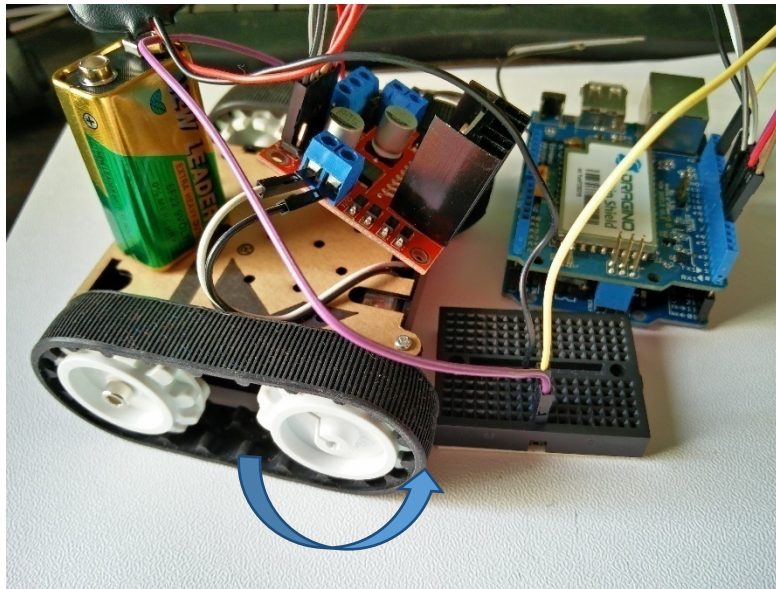


Fig 4.2: Motor 1 Forward.

Pressing 3 at serial monitor motor 1 goes backward

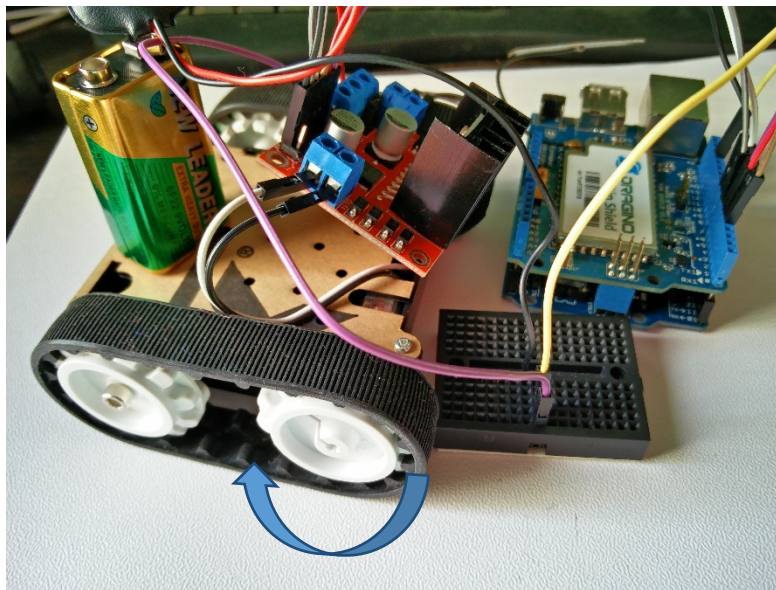


Fig 4.3: Motor 1 Backward

Pressing 4 at serial monitor motor 2 goes forward



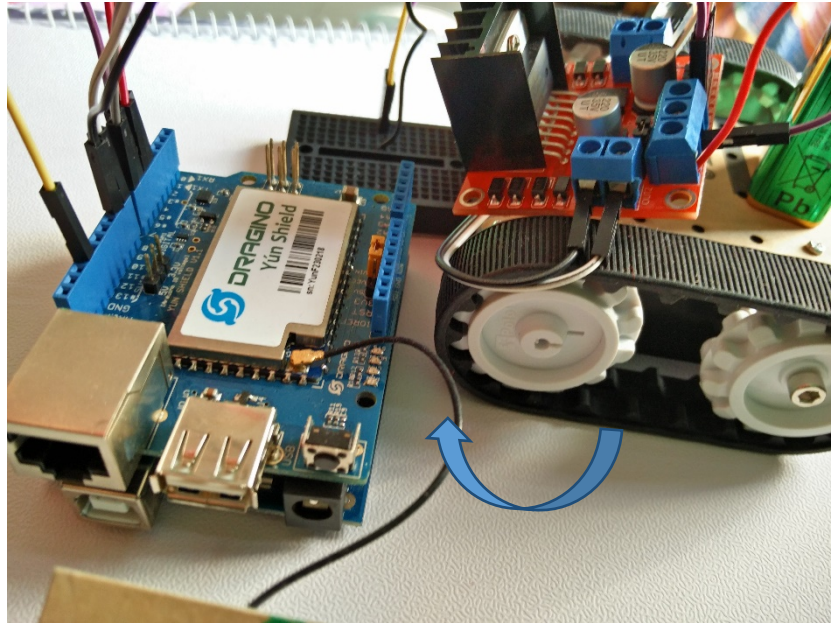


Fig 4.4: Motor 2 Forward

Pressing 5 at serial monitor motor 2 goes backward

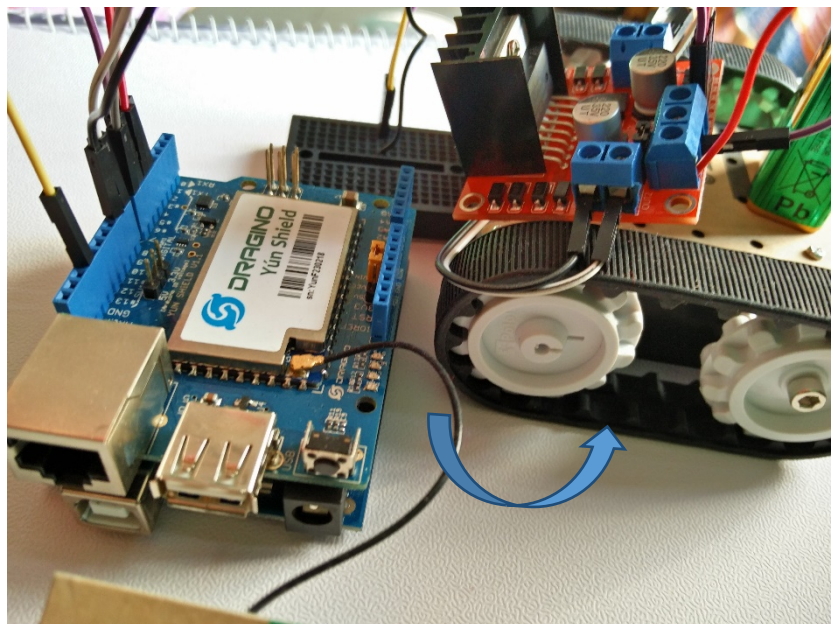


Fig 4.5: Motor 2 Backward

In Fig 4.2 when the serial monitor receives case 1 with PWM value set to 80, Motor 1 rotates at clockwise.

When the serial monitor receives case 2 with PWM value set to 0, Motor 1 stops as the PWM value is 0.

In Fig 4.3 when the serial monitor receives case 3 with PWM value set to 80, Motor 1 rotates counter clockwise.

In Fig 4.4 when the serial monitor receives case 4 with PWM value set to 80, Motor 2 rotates at clockwise.

When the serial monitor receives case 5 with PWM value set to 0, Motor 2 stops.

In Fig 4.5 when the serial monitor receives case 6 with PWM value set to 80, Motor 2 rotates counter clockwise.

The work of the remote web server will be to use Rest function on Arduino library to wait for the get signals (case 1-case 5)

# **Chapter Five**

## **Conclusion & Further Research**

## 5.1 Conclusion

Technology now a day has taken on a whole new meaning, we have incredibly “smart” smart phones, new ways of creating energy, and now autonomous cars. The WiFi controlled car is a mind blowing innovation that could change how we live and transport ourselves how our home surveillance system works. Although people don’t fully trust artificial intelligence we tried to make a product that is truly flawless. We believe, that in a couple of decades, our roads and high ways will be filled with the autonomous car.

## 5.2 Future Work

Self-driving cars are a rapidly evolving technology which only a few years ago was still considered science fiction. In such a dynamic context, quick intuitions can be very misleading and misconceptions about the technology, its impact, and the nature of the innovation process abound.

This paper talks about the potential benefits, legal and ethical issues, security concerns, social problems, further required research of finding the way of making a Self-driving car in cheap budget. The project itself is very simple and environment friendly.

## 5.3 References

- [1] <https://www.arduino.cc/en/Main/ArduinoBoardUno>
- [2] <http://www.whispers.co.jp/wp-content/uploads/2014/11/Arduino-uno-Pinout-1.png>
- [3] <https://www.arduino.cc/en/Main/ArduinoBoardYun>
- [4] <http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/>
- [5] <https://www.arduino.cc/en/Tutorial/DueMotorShieldDC>
- [6] <http://blog.whatgeek.com.pt/arduino/l298-dual-h-bridge-motor-driver/>

# List of Figures

<b>Figure No.</b>	<b>Figure Name</b>	<b>Page</b>
<b>2.1</b>	Arduino UNO	<b>13</b>
<b>2.2</b>	A typical Arduino UNO board showing all of its I/O ports	<b>15</b>
<b>2.3</b>	Arduino Pin Mapping	<b>16</b>
<b>2.4</b>	Arduino IDE Window	<b>17</b>
<b>2.5</b>	Arduino YUN Shield	<b>18</b>
<b>2.6</b>	YUN arduino and linux environment	<b>19</b>
<b>2.7</b>	YUN Indicator LED	<b>22</b>
<b>2.8</b>	Yun Reset buttons	<b>23</b>
<b>2.9</b>	Motor Drive PWM	<b>26</b>
<b>2.10</b>	L298N Motor drive connection	<b>27</b>
<b>2.11</b>	Motor drive Circuit	<b>29</b>
<b>3.1</b>	System block diagram	<b>31</b>
<b>3.2</b>	Arduino UNO and Yun Connection	<b>32</b>
<b>3.3</b>	L298N Motor drive	<b>33</b>
<b>3.4</b>	Micro Metal Gearmotor	<b>34</b>
<b>3.5</b>	Chassis	<b>34</b>
<b>3.6</b>	Completed system diagram	<b>35</b>
<b>3.7</b>	Software design IDE	<b>36</b>
<b>3.8</b>	System control flow chart	<b>41</b>
<b>3.9</b>	Software Upload IDE	<b>42</b>
<b>3.10</b>	Serial Monitor	<b>43</b>
<b>3.11</b>	System Connection Scematic Diagram	<b>44</b>
<b>4.1</b>	WiFi Controlled Robot (graphical representation)	<b>46</b>
<b>4.2</b>	Motor 1 Forward	<b>47</b>
<b>4.3</b>	Motor 1 Backward	<b>47</b>
<b>4.4</b>	Motor 2 Forward	<b>48</b>
<b>4.5</b>	Motor 2 Backward	<b>48</b>