

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Palestine Polytechnic University

College of Engineering

Electrical Engineering Department

Graduation Project

**Designing Robot to be Controlling by Microcontroller Via
Wireless Network with 3D-Dimension Glasses Technique**

"DISCOVERY ROBOT"

Project Team:

Lieth R. Tomar

Khaldoun H. Jabareen

Project Supervisor:

Dr. Raed Amro

Hebron _ Palestine

2017-2018

جامعة بوليتكنك فلسطين

كلية الهندسة

دائرة الهندسة الكهربائية

اسم المشروع

**Designing Robot to be Controlling by Microcontroller Via
Wireless Network with 3D-Dimension Glasses Technique**

"DISCOVERY ROBOT"

فريق العمل:

ليث ربحي طومار - 145482

خلدون حسين جبارين - 135390

بناء على توجيهات المشرف على المشروع وبموافقة جميع أعضاء اللجنة
المتحنة، تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية للوفاء بمتطلبات
الدائرة لدرجة البكالوريوس.

توقيع المشرف

توقيع اللجنة المتحنة

توقيع رئيس الدائرة

شكر و تقدير

"كن عالما .. فإن لم تستطع فكن متعلما ، فإن لم تستطع فأحب العلماء"

"بعد رحلة بحث و جهد و اجتهاد تكلفت بإنجاز هذا المشروع" ، نحمد الله عز وجل على نعمه التي من بها علينا فهو العلي القدير ، كما لا يسعنا إلا أن نخص بأسمى عبارات الشكر و التقدير لـدكتور "رائد عمرو" لما قدمه لنا من جهد و نصح و معرفة طيلة انجاز هذا المشروع .

كما نتقدم بالشكر الجزيل لكل من أسهم في تقديم يد العون لإنجاز هذا المشروع، و نخص بالذكر أستاذنا الكرام الذين منحونا علمهم ، و الأستاذة القائمين على عمادة و إدارة كلية الهندسة ، كما لا ننسى أن نتقدم بأرقى و أئمن عبارات الشكر و العرفان إلى القائمين على الجامعة و نخص بذكر ادارة الجامعة و كل العاملين بها .

والى الذين كانوا عوننا لنا في بحثنا هذا، إلى من زرعوا التفاؤل في دربنا وقدموا لنا المساعدات والتسهيلات والمعلومات ، فلمن منا كل الشكر، وأخص زملائنا وادقائنا الذين أسهموا بشكل وثير في تشجيعنا أثناء انجاز المشروع . أما الشكر الذي من النوع الخاص فنحن نتوجه بالشكر أيضا إلى كل زملائنا الخريجين، فلولا جودهم لما أحسنا بمتعة العمل و حلاوة البحث وروح التحدي، و لما وطننا إلى ما وطننا إليه فلمن منا كل الشكر.

الإهداء

بسم الله الرحمن الرحيم

(قل اعملوا فسيرى الله عملكم ورسوله والمؤمنون)

صدق الله العظيم

إلهي لا يطيب لي الليل إلا بشكرك ولا يطيب لي النهار إلا بطاعتك .. ولا تطيب لي اللحظات إلا بشكرك .. ولا تطيب لي الآخرة إلا بعفوك .. ولا تطيب لي الجنة إلا برويتك إلى من بلغ الرسالة وأدى الأمانة .. ونصح الأمة .. إلى نبي الرحمة ونور العالمين "سيدنا محمد صلى الله عليه وسلم"

إلى مسرى الرسول عليه السلام وعاصمة الدولة الفلسطينية الأبدية "القدس"

إلى أرواح شهدائنا الأبرار، إلى أسرارنا البواسل نهدي عملنا المتواضع

إلى من كلفه الله بالصيبة والوقار .. إلى من علمني العطاء بدون انتظار .. إلى من أحمل اسمه بكل افتخار .. أرجو من الله أن يمد يدي عمرك لتري ثماراً قد حان قطافها بعد طول انتظار وستبقى كلماتك نجوم أهدني بها اليوم وفي الغد

والى الأبدي "والدي العزيز"

إلى ملاكي في الحياة .. إلى معنى الحب وإلى معنى العنان و التفاني إلى

بسمة الحياة وسر الوجود

إلى من كان دعائها سر نجائي وحنانها بلسم جراحي إلى أجلي الحبايب

"أمي الحبيبة"

Abstract:

Under the scientific development and increase scientific researches, there must be access to places that humans can't reach and explore and study them, so this robot is designed to access places and transmit the image of these places using a 3d technology and the dimensions of these places, and are Control the robot using wireless technology via the remote control, sending and receiving data over the wireless network and processing data using the Microcontroller (Arduino).

ملخص :

في ظل التطور العلمي وازدياد الأبحاث العلمية لا بد من الوصول الى الأماكن التي لا يستطيع البشر الوصول إليها واستكشاف ما حولها ودراستها لذلك تم تصميم هذا الروبوت ليتمكننا من الوصول الى الأماكن ونقل صورة هذه الأماكن باستخدام تقنيه ثلاثيه الأبعاد وإبعاد هذه الأماكن ، ويتم التحكم في الروبوت بتقنيه لاسلكية باستخدام جهاز التحكم عن بعد وإرسال واستقبال البيانات عبر الشبكة اللاسلكية ومعالجه البيانات باستخدام المعالج الدقيق (الاردوينو) .

Table of Contents

<i>First Pages</i>		<i>Page Number</i>
<i>Book Interface</i>		<i>I</i>
<i>Project Evaluation</i>		<i>II</i>
<i>Thanks and Appreciation</i>		<i>III</i>
<i>Dedication</i>		<i>IV</i>
<i>Abstract (English)</i>		<i>V</i>
<i>Abstract (Arabic)</i>		<i>VI</i>
<i>Table of Contents</i>		<i>VII</i>
Chapter One	Introduction	1
<i>1.1</i>	<i>Overview</i>	1
<i>1.2</i>	<i>Project Objectives</i>	1
<i>1.3</i>	<i>Project Component Table</i>	2
<i>1.4</i>	<i>Block Diagram</i>	3
<i>1.5</i>	<i>Economical Study</i>	4
<i>1.6</i>	<i>Tasks Time Schedule</i>	5
<i>1.7</i>	<i>Advantages And Disadvantages</i>	6
<i>1.8</i>	<i>Review of Literature</i>	6
Chapter Two	Robot Structure	7
<i>2.1</i>	<i>Microcontroller (Arduino Mega)</i>	8
<i>2.1.1</i>	<i>Overview</i>	8
<i>2.1.2</i>	<i>Power</i>	8
<i>2.1.3</i>	<i>Input and Output</i>	9
<i>2.1.4</i>	<i>Memory</i>	9
<i>2.1.5</i>	<i>Technical Specifications</i>	10
<i>2.2</i>	<i>Wireless module</i>	11
<i>2.2.1</i>	<i>Overview</i>	11

2.2.2	<i>Wireless module nRF24L01</i>	11
2.2.3	<i>Signal Range</i>	11
2.2.4	<i>Network Configuration</i>	12
2.2.5	<i>Transmitting Payload (TX)</i>	13
2.2.6	<i>Receive Payload (RX)</i>	13
2.2.7	<i>Pin Functions</i>	14
2.3	<i>Motor Driver (H-Bridge)</i>	15
2.3.1	<i>Overview</i>	15
2.3.2	<i>Motor Driver L293D</i>	15
2.3.3	<i>Power Supply</i>	16
2.3.4	<i>Operation</i>	16
2.4	<i>DC Motor</i>	17
2.4.1	<i>Overview</i>	17
2.4.2	<i>Motor Control</i>	17
2.4.3	<i>Motor Poles</i>	18
2.4.4	<i>Torque and Speed of DC Motor</i>	18
2.4.5	<i>DC Motor Specifications</i>	20
2.5.6	<i>Specifications Pins of H-Bridge</i>	20
2.5	<i>Servo motor</i>	21
2.5.1	<i>Overview</i>	21
2.5.2	<i>Servo motor SG-90</i>	21
2.5.3	<i>Pine Configuration</i>	22
2.5.4	<i>Characteristics and features of servo motor</i>	22
2.6	<i>Ultrasound sensor</i>	23
2.6.1	<i>Overview</i>	23
2.6.2	<i>Effectiveness of Ultrasound</i>	23
2.6.3	<i>Timing Diagram</i>	23
2.6.4	<i>Sensor Pine Configuration</i>	24
2.7	<i>FPV Camera</i>	25
2.7.1	<i>Overview</i>	25
2.7.2	<i>Promark VR Application</i>	25
2.8	<i>Battery</i>	26
2.8.1	<i>Lithium Battery for microcontroller Unit</i>	26
2.8.2	<i>Table of Loads</i>	26
2.8.3	<i>Specifications</i>	26
2.8.4	<i>Lithium Battery for H-Bridge</i>	27

2.8.5	<i>Table of Loads</i>	27
2.8.6	<i>Specifications</i>	27
2.9	<i>Robot Body</i>	28
2.10	<i>Circuit Connecting Devices with microcontroller</i>	29
2.10.1	<i>Circuit Diagram</i>	29
2.10.2	<i>Electrical Circuit</i>	30
Chapter Three	<i>Remote Control Module</i>	31
3.1	<i>Microcontroller (Arduino Uno)</i>	32
3.1.1	<i>Overview</i>	32
3.1.2	<i>Power</i>	32
3.1.3	<i>Input and Output</i>	33
3.1.4	<i>Memory</i>	34
3.2.1	<i>Technical Specifications</i>	34
3.2	<i>Wireless module</i>	35
3.2.1	<i>Overview</i>	35
3.2.2	<i>Wireless module nRF24L01</i>	35
3.2.3	<i>Signal Range</i>	35
3.2.4	<i>Network Configuration</i>	36
3.2.5	<i>Transmitting Payload (TX)</i>	37
3.2.6	<i>Receive Payload (RX)</i>	37
3.2.7	<i>Pin Functions</i>	38
3.3	<i>Joysticks</i>	39
3.3.1	<i>Overview</i>	39
3.3.2	<i>Description Joystick</i>	39
3.3.3	<i>Joystick control</i>	40
3.3.4	<i>Pin Functions</i>	40
3.4	<i>Liquid Crystal Display (LCD)</i>	41
3.4.1	<i>Overview</i>	41
3.4.2	<i>LCD LCM1602 IIC</i>	41
3.4.3	<i>LCM Specification</i>	42
3.4.4	<i>Pin Functions</i>	42
2.5	<i>Lithium Battery</i>	43
3.5.1	<i>Overview</i>	43

3.5.2	<i>Table of Loads</i>	43
3.5.3	<i>Specifications of Battery</i>	43
3.6	<i>Remote Body</i>	44
3.7	<i>Circuit Connecting Devices with microcontroller</i>	45
3.7.1	<i>Circuit Diagram</i>	45
3.7.2	<i>Electrical Circuit</i>	46
Chapter Four	<i>3D Glasses Module</i>	47
4.1	<i>Microcontroller (Arduino Nano)</i>	48
4.1.1	<i>Overview</i>	48
4.1.2	<i>Power</i>	48
4.1.3	<i>Input and Output</i>	48
4.1.4	<i>Memory</i>	49
4.1.5	<i>Technical Specifications</i>	50
4.2	<i>Wireless Module</i>	51
4.2.1	<i>Overview</i>	51
4.2.2	<i>Wireless module nRF24L01</i>	51
4.2.3	<i>Signal Range</i>	51
4.2.4	<i>Network Configuration</i>	52
4.2.5	<i>Transmitting Payload (TX)</i>	53
4.2.6	<i>Receive Payload (RX)</i>	53
4.2.7	<i>Pin Functions</i>	54
4.3	<i>MPU Sensor</i>	55
4.3.1	<i>Overview</i>	55
4.3.2	<i>MPU-6050</i>	55
4.3.3	<i>Pin Configuration</i>	56

<i>4.4</i>	<i>Battery</i>	<i>57</i>
<i>4.4.1</i>	<i>Overview</i>	<i>57</i>
<i>4.4.2</i>	<i>Table of Loads</i>	<i>57</i>
<i>4.4.3</i>	<i>Specifications</i>	<i>57</i>
<i>4.5</i>	<i>VR Glasses</i>	<i>58</i>
<i>4.5.1</i>	<i>Overview</i>	<i>58</i>
<i>4.5.2</i>	<i>Siswoo VR Glasses</i>	<i>58</i>
<i>4.6</i>	<i>Circuit Connecting Devices with microcontroller</i>	<i>59</i>
<i>4.6.1</i>	<i>Circuit Diagram</i>	<i>59</i>
<i>4.6.2</i>	<i>Electrical Circuit</i>	<i>60</i>
Chapter Five	Programming	61
<i>5.1</i>	<i>Arduino Program</i>	<i>61</i>
<i>5.1.1</i>	<i>Overview</i>	<i>61</i>
<i>5.1.2</i>	<i>Structure</i>	<i>61</i>
<i>5.1.3</i>	<i>Sub-Program</i>	<i>61</i>
<i>5.2</i>	<i>Robot Code</i>	<i>62</i>
<i>5.2.1</i>	<i>Code Description</i>	<i>62</i>
<i>5.2.2</i>	<i>Robot Code</i>	<i>62</i>
<i>5.2.3</i>	<i>Flow Chart</i>	<i>66</i>
<i>5.3</i>	<i>Remote Control Code</i>	<i>67</i>
<i>5.3.1</i>	<i>Code Description</i>	<i>67</i>
<i>5.3.2</i>	<i>Remote Code</i>	<i>67</i>
<i>5.3.3</i>	<i>Flow Chart</i>	<i>70</i>
<i>5.4</i>	<i>3D-Glasses Code</i>	<i>71</i>

<i>5.4.1</i>	<i>Code Description</i>	<i>71</i>
<i>5.4.2</i>	<i>3D-Glasses Code</i>	<i>71</i>
<i>5.4.3</i>	<i>Flow Chart</i>	
	<i>Conclusion</i>	<i>77</i>
	<i>References</i>	<i>78</i>

Chapter One

Introduction

1.1 Overview

Robots are increasingly applied in automation and production fields, a further application field is currently discovered in detecting and identifying of objects and obstacles in dangerous areas, clear advancement in control methods and technologies make such application possible. Using modern technologies such as wireless and remote control. And see that what around us using a three-dimensional technique.

Our robot will be controlled by Arduino which is an electronic development board consisting of an electronic circuit with a microcontroller programmed by computer, designed to facilitate the use of interactive electronics in multidisciplinary projects. Arduino is mainly used in the design of interactive electronic projects or projects aimed at building different environmental sensors.












Arduino can be connected to various programs on the PC, and its programming is based on C language and is one of the easiest programming languages used to write microcontroller programs.

1.2 Project Objectives

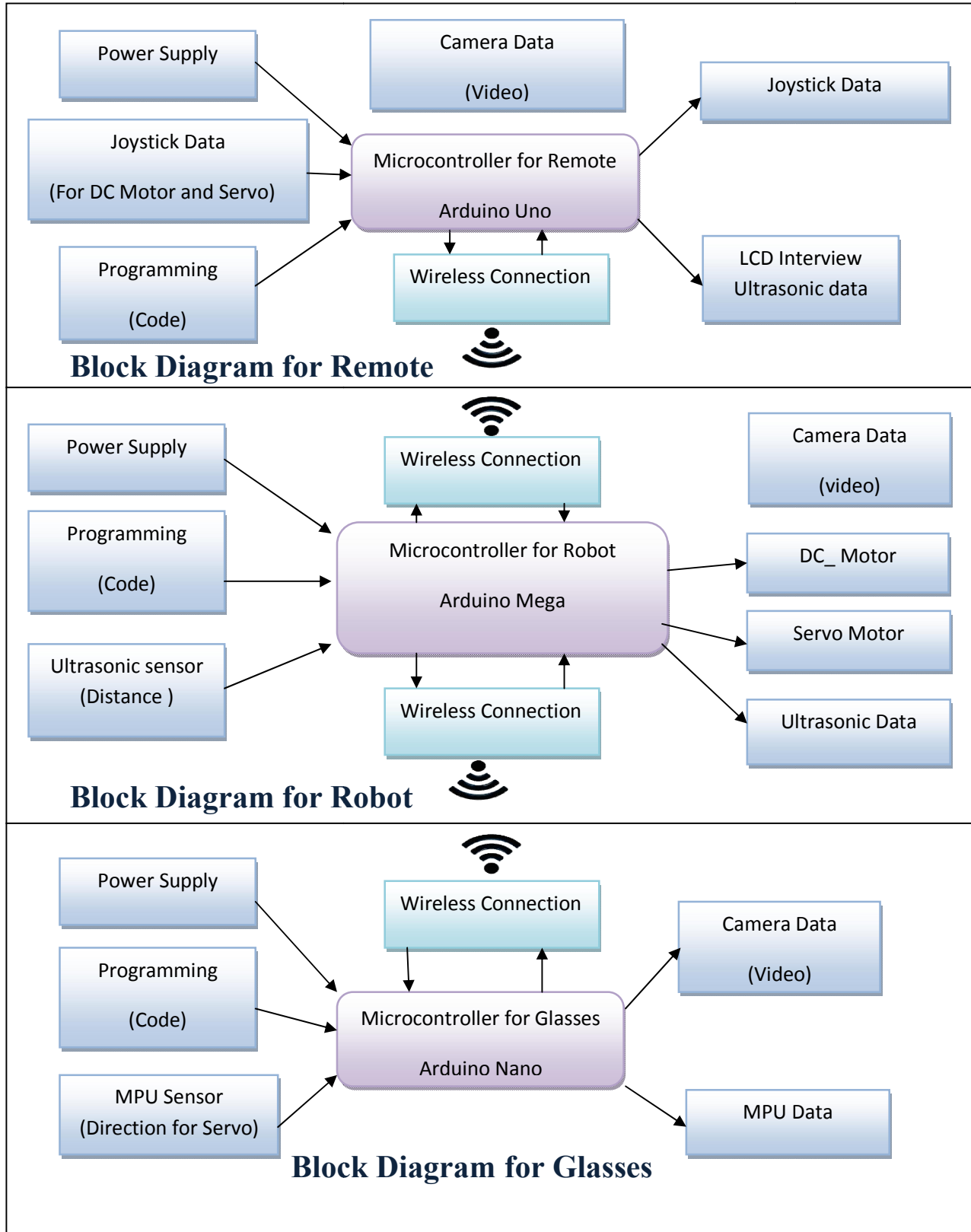
- Explore what's around us by using a 3-D technique.
- Using Smartphone to control the applications.
- Using wireless technology for remote control.
- Send specific commands and read data.
- Add new techniques using programming skills.
- Designing a robot and connecting its parts and programming.

1.3 Project Components Table

This system that contains sensors, control systems, power supplies and software all working together to perform a task. Designing, building, programming and testing robots is a combination of physics, mechanical engineering, electrical engineering, structural engineering, mathematics and computing.

Part	Photo	Brief
Arduino		A microcontroller programmed by computer, designed to facilitate the use of interactive electronics in multidisciplinary projects.
Ultrasonic Sensor		A device for measuring the distance using ultrasound
Servo Motor		rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity
FPV Camera		Device that transfers the image
DC Motor		The motor moves the robot in all directions
H_Bridge		Device of separation of the control circuit from the power circuit of the motor
NRF24L01		Wireless Transceiver Module
LCD		Display Screen
Battery		Power supply
MPU		A sensor to control the direction of the camera
Joystick		device controlling a direction and angles of parts the robot

1.4 Block Diagram



1.8 Economical Study

Devices	Number of pieces needed	Price per Piece /₱
DC Motor	2	200
Car Toy	1	200
3D Glasses	1	100
Servo Motor	2	50
H_Bridge	1	50
Battery	4	200
Arduino Nano	1	45
Arduino Uno	1	75
Arduino mega	1	190
FPV Camera	1	440
Ultrasonic Sensor	1	50
NRF24L01	3	300
MPU	1	45
JOYSTICK	2	70
LCD	1	100
Wire and board	40	160
Transportation	6	450
Total		2725 ₱

Table1.1 Project Hardware Cost

1.9 Tasks Time Schedule

<i>#. Week</i>	1	2	3	4	5	6	7	8	9	10	11	12
Buy the Part and Devices of Robot												
Testing the Parts												
Programming												
Building a Robot												
Testing the Robot												
Running a Robot												
Research												
Writing Report												

Table1.2 Distribution of tasks Schedule

1.5 Advantages and Disadvantages

The advantages of this robot, many of which are very low cost with many features, are also small-sized, easy to control, easy programming, and saves time with better accuracy and performance. And the ability to explore for a number of hours without stopping and see what is around us very accurately.

There can also be some disadvantages, such as being unable to work in harsh conditions or working in places that do not fit the nature such as rugged terrain or work in a water-containing environment.

1.6 Review of Literature

There are some Researchers people who have tried to use phones to control some remote objects. Perhaps the most prominent of these works robot Explorer (MER-B) Mars Exploration Rover, Which was sent by NASA on Mars in the task of searching for water on the surface and study rocks and breeding, with wide-angle panoramic camera, a navigational and positioning on Mars, Thermometer spectrometer, identifies stones and soil type of importance, and can analyze samples and determine methods of formation [1].

And from scientific devices a spectrometer that works with the influence of mouspauer can examine samples containing iron, an X-ray spectrometer that works with alpha particles. It can analyze sample compounds and determine the proportions of their components. Magnet picks magnetic samples, a microscope that depicts stones and soil.

Other researchers have tried to design a small toy car and control it by placing special numbers on the phone board using the software. Including WebBot, which gave a detailed explanation of the design and applications of robots and control [2].

Chapter two

Robot Structure



2.1 Microcontroller (Arduino Mega)

2.1.1 Overview

The Arduino Mega is a microcontroller board based. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, and a reset button. It contains everything needed to support the microcontroller; board can be powered via the external power supply using battery [3].

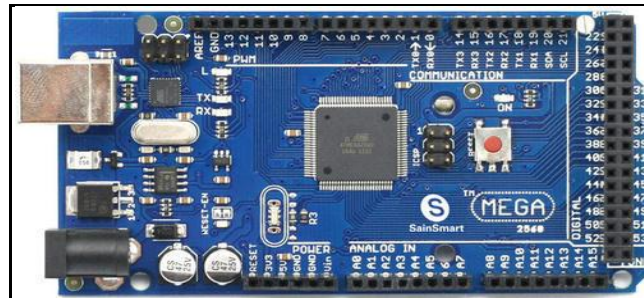


Figure 2.1

2.1.2 Power

The Arduino Mega can be powered via external power supply. Leads from a battery can be inserted in the GND and VIN pin headers of the power connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable.

If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V: The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V: A 3.3 volt. Maximum current draw is 50 mA.
- GND: Ground pins.

2.1.3 Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using [pinMode()], [digitalWrite()], and [digitalRead()] functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA. The Mega has 16 analog inputs, each of which provides 10 bits of resolution. By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and [analogReference()] function.

2.1.4 Memory

The Mega has 128 KB of flash memory for storing code of which 4 KB is used for the boot loader, 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

2.1.5 Technical Specifications

<i>Microcontroller</i>	<i>ATmega2560</i>
<i>Operating Voltage</i>	<i>5V</i>
<i>Input Voltage (recommended)</i>	<i>7-12V</i>
<i>Input Voltage (limit)</i>	<i>6-20V</i>
<i>Digital I/O Pins</i>	<i>54 of which 15 provide PWM output</i>
<i>Analog Input Pins</i>	<i>16</i>
<i>DC Current per I/O Pin</i>	<i>20 mA</i>
<i>DC Current for 3.3V Pin</i>	<i>50 mA</i>
<i>Flash Memory</i>	<i>256 KB of which 8 KB used by boot loader</i>
<i>SRAM</i>	<i>8 KB</i>
<i>EEPROM</i>	<i>4 KB</i>
<i>Clock Speed</i>	<i>16 MHz</i>
<i>LED_BUILTIN</i>	<i>13</i>
<i>Length</i>	<i>101.52 mm</i>
<i>Width</i>	<i>53.3 mm</i>
<i>Weight</i>	<i>37 g</i>

Table 2.1

2.2 Wireless module

2.2.1 Overview

A Wireless module that sends and receives commands in a wireless way and is connected to different Arduino types to execute incoming and future orders from different controllers.

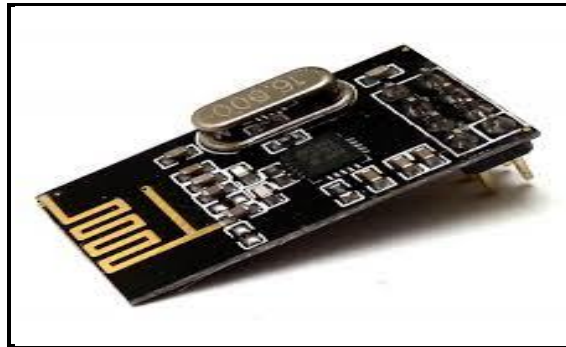


Figure 2.2

2.2.2 Wireless module nRF24L01

Modem of NRF works on 2.4GHz frequency and has 100m range (at best). You can communicate with different baud rates: 250 kbps, 1 or 2 Mbps. High baud rate allows users even to send video material in 720p resolution. During receive power consumption is around 12mA and during transmission 11mA. NRF can operate on voltages between 1.9 and 3.6V. The input pins tolerate the 5V logic, so without any problems this module can be connected to the most popular development boards like Arduino. Most models of Arduino (Uno, Mega, Nano and more) are working with these logic levels [4].

2.2.3 Signal Range

The signal range of nRF24L01 Module if used in open space and with lower baud rate its range can reach up to 100 meter.

2.2.4 Network Configuration

An nRF24L01 configured as primary RX (PRX) will be able to receive data through 6 different data pipes that have a unique address but share the same frequency channel. This means that up to 6 different nRF24L01 configured as primary TX (PTX) can communicate with one nRF24L01 configured as PRX, and the nRF24L01 configured as PRX will be able to distinguish between them. Data pipe 0 has a unique 40-bit configurable address.

Each of data pipes 1-5 has an 8-bit unique address and shares the 32 most significant address bits. All data pipes can perform full Enhanced functionality. nRF24L01 will use the data pipe address when acknowledging a received packet. This means that nRF24L01 will transmit with the same address as it receives payload at. In the PTX device, data pipe 0 is used to receive the acknowledge, and therefore the receive address for data pipe 0 has to be equal to the transmit address to be able to receive the acknowledge.

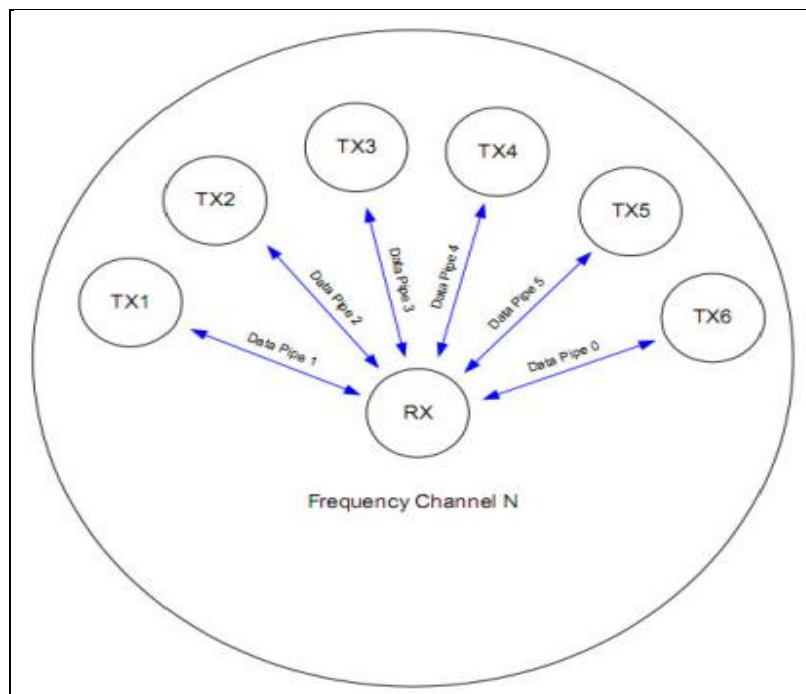


Figure2.3

2.2.5 Transmitting Payload (TX)

- The configuration bit PRIM_RX has to be low, address settings have to be performed for the TX device and the RX device:

TX device: TX_ADDR = 0xB3B4B5B605

RX device: RX_ADDR = 0xB3B4B5B605

- A high pulse on CE starts the transmission. The minimum pulse width on CE is 10 μ s.
- nRF24L01 Shock Burst:

* Radio is powered up

*16 MHz internal clock is started.

*RF packet is completed (see the packet description)

* Data is transmitted at high speed (1 Mbps or 2 Mbps configured by MCU).

- The IRQ pin will be active when MAX_RT or TX_DS is high. To turn off the IRQ pin, the interrupt source must be reset by writing to the status register (see Interrupt chapter). If no acknowledgement is received for a packet after the maximum number of retries, no further packets can be sent before the MAX_RX interrupt is cleared.
- The device goes into Standby-I mode if CE is low. Otherwise next payload in TXFIFO will be sent. If TX FIFO is empty and CE is still high, the device will enter Standby -II mode.
- If the device is in Standby-II mode, it will go to Standby-I mode immediately if CE is set low.

2.2.6 Receive Payload (RX)

- RX is selected by setting the PRIM_RX bit in the configuration register to high, all data pipes that shall receive data must be enabled (EN_RXADDR register), auto acknowledgement for all pipes running Enhanced ShockBurst has to be enabled (EN_AA register), and the correct payload widths must be set (RX_PW_Px registers).

- Addresses have to be set up as described in item 2 in the Enhanced ShockBurst transmit payload chapter above.
- Active RX mode is started by setting CE high.
- After 130µs nRF24L01 is monitoring the air for incoming communication.
- When a valid packet has been received (matching address and correct CRC), the payload is stored in the RX-FIFO, and the RX_DR bit in status register is set high. The IRQ pin will be active when RX_DR is high. RX_P_NO in status register will indicate what data pipe the payload has been received in.
- If auto acknowledgement is enabled, an acknowledgement is sent back.
- MCU sets the CE pin low to enter Standby-I mode (low current mode).
- MCU can clock out the payload data at a suitable rate via the SPI interface.
- The device is now ready for entering TX or RX mode or power down mode.

2.2.7 Pin Functions

<i>Pin</i>	<i>Name</i>	<i>Description</i>	<i>Pin function</i>	<i>Description</i>
1	CE		Digital Input	Chip Enable Activates RX or TX mode
2	CSN		Digital Input	SPI Chip Select
3	SCK		Digital Input	SPI Clock
4	MOSI		Digital Input	SPI Slave Data Input
5	MISO		Digital Input	SPI Slave Data Output, with tri-state option
6	IRQ		Digital Input	Maskable interrupt pin
7	VDD		Power	Power Supply (+3V DC)
8	VSS		Power	Ground (0V)

Table 2.2

2.3 Motor Driver (H-Bridge)

2.3.1 Overview

Motor Driver is an electronic device use to control DC motors such as servo motor and stepper motor. It serves as a link between microcontrollers and motor, it can control the speed and direction of DC motors and the motion angle of servo motors and stepper motors by microcontroller.

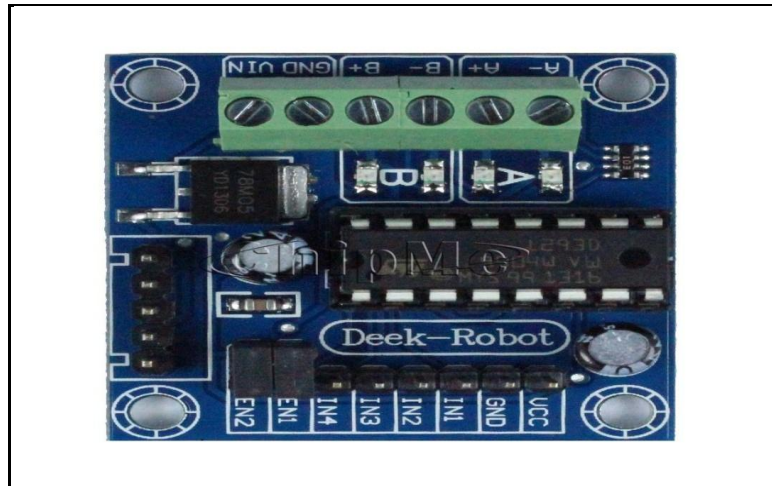


Figure 2.4

2.3.2 Motor Driver L293D

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoids, DC and stepping motors) and switching power transistors .To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included. This device is suitable for use in switching applications at frequencies up to 5 kHz [5].

2.3.3 Power Supply

Supply the power either through the Arduino board or the power header of the Motor Shield. Only if the Arduino board and the Shield use a separate power supply, the jumper must be removed, the device can be supplied from a Battery source from (6-12) volts.

2.3.4 Operation

The Motor Driver arrangement is generally used to reverse the polarity/direction of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit. The following table 2.2 summarizes operation, with S1-S4 corresponding to the (figure 2.5).

S1	S2	S3	S4	Result
1	0	0	1	Motor Moves Right
0	1	1	0	Motor moves left
0	1	0	1	Motor Forward
1	0	1	0	Motor Backward
0	0	0	0	Motors Off
1	1	0	0	Short circuit
0	0	1	1	Short circuit

Table 2.3

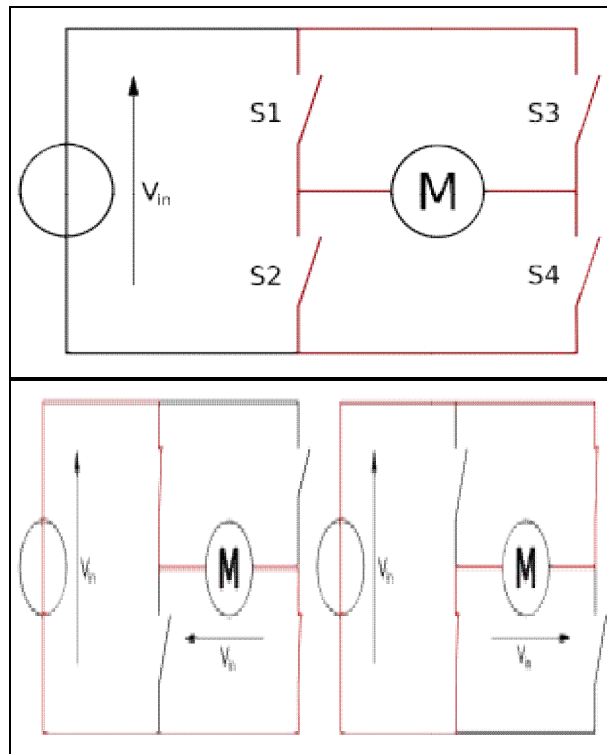


Figure 2.5

2.4 DC Motor

2.4.1 Overview

A Brushed DC Motor is an internally commutated electric motor designed to be run from a direct current power source. Brushed DC Motors can be varied in speed by changing the operating voltage or the strength of the magnetic field. Depending on the connections of the field to the power supply, the speed and torque characteristics of a brushed motor can be altered to provide steady speed or speed inversely proportional to the mechanical load.



Figure 2.6

2.4.2 Motor Control

The rotational speed of a DC motor is proportional to the EMF in its coil (= the voltage applied to it minus voltage lost on its resistance), and the torque is proportional to the current. The direction of a wound field DC motor can be changed by reversing either the field or armature connections but not both. But to control the robot at constant speed and constant torque so the motor driver was used, as the direction of rotation is controlled by the motor driver through the microcontroller signal, which is controlled by programming.

2.4.3 Motor Poles

When a current passes through the coil wound around a soft iron core, the side of the positive pole is acted upon by an upwards force, while the other side is acted upon by a downward force. According to Fleming's left hand rule, the forces cause a turning effect on the coil, making it rotate. To make the motor rotate in a constant direction, "direct current" commutator make the current reverse in direction every half a cycle (in a two-pole motor) thus causing the motor to continue to rotate in the same direction[6].

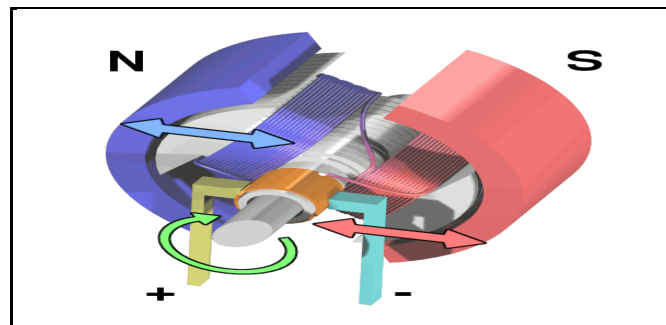


Figure 2.7

2.4.4 Torque and Speed of DC Motor

Basic properties:

- E_b , induced or counter EMF (V)
- I_a , armature current (A)
- 0
- k_n , speed equation constant
- k_T , torque equation constant
- n , armature frequency (rpm)
- R_m , motor resistance (Ω)
- T , motor torque (Nm)
- V_m , motor input voltage (V)
- Φ , machine's total flux (Wb)

Counter EMF equation

The DC motor's counter emf is proportional to the product of the machine's total flux strength and armature speed:

$$E_b = k_b \Phi n$$

Voltage balance equation

The DC motor's input voltage must overcome the counter emf as well as the voltage drop created by the armature current across the motor resistance, that is, the combined resistance across the brushes, armature winding and series field winding, if any:

$$V_m = E_b + R_m I_a$$

Torque equation

The DC motor's torque is proportional to the product of the armature current and the machine's total flux strength:

$$T = \frac{1}{2\pi} k_b I_a \Phi$$

$$T = k_T I_a \Phi$$

Speed equation

$$n = \frac{E_b}{k_b \Phi}$$

$$V_m = E_m + R_m I_m$$

$$n = \frac{V_m - R_m I_a}{k_b \Phi}$$

$$n = k_n \frac{V_m - R_m I_a}{\Phi}$$

2.4.5 DC Motor Specifications

<i>Type</i>	<i>Gear Motor</i>
<i>Voltage(V)</i>	<i>24V</i>
<i>Current</i>	<i>0.37A</i>
<i>Speed(RPM)</i>	<i>200rpm</i>
<i>Speed(RPM) at 12V</i>	<i>0100rpm</i>
<i>Torque</i>	<i>3Kg.cm MAX</i>
<i>Output Power</i>	<i>2.7W MAX</i>

Table2.4

2.4.6 Specifications Pins of H-Bridge

Pin	Name	Description
1	<i>Output-O1</i>	<i>Motor A+ lead out</i>
2	<i>Output-O2</i>	<i>Motor A- lead out</i>
3	<i>Output -O3</i>	<i>Motor B+ lead out</i>
4	<i>Output -O4</i>	<i>Motor B- lead out</i>
5	<i>GND</i>	<i>Power ground</i>
6	<i>+5</i>	<i>5V Power supply</i>
7	<i>Input-In1</i>	<i>Enable Motor A+</i>
8	<i>Input-In2</i>	<i>Enable Motor A-</i>
9	<i>Input-In3</i>	<i>Enable Motor B+</i>
10	<i>Input-In4</i>	<i>Enable Motor B-</i>

Table 2.5

2.5 Servo motor

2.5.1 Overview

A servo motor is a type of motor that is often used in robotics. The position of the motor can be controlled very precisely . a servo motor requires a signal to tell the motor how far to turn. Most servo motors have a 3 wire interface, One for input voltage, one goes to ground and one for control signal. This is called Pulse Width Modulation, often shortened to PWM. The value of the angle or direction depends on the value of pwm, which determines the angle of the servo motor, the angle between zero to 180 degrees.



Figure2.8

2.5.2 Servo motor SG-90

In this project the servo motor use to direct the camera's arm, where the direction of the servo motor or camera is controlled by the joystick or by MPU, the servo motors were used to turn the camera right, left, and up, down.

There are lots of servo motors available in the market and each one has its own specialty and applications. Most of the hobby Servo motors operates from 4.8V to 6.5V, the higher the voltage higher the torque we can achieve, but most commonly they are operated at + 5V. Almost all hobby servo motors can rotate only from 0 ° to 180 ° due to their gear arrangement, Next comes the most important parameter[7].

Which is the torque at which the motor operates, are many choices here but the commonly available one is the 2.5kg / cm torque which comes with the Motor. This 2.5kg / cm torque means that the motor can pull a weight of 2.5kg when it is suspended at a distance of 1cm.

2.5.3 Pine Configuration

<i>Pin Number</i>	<i>Pin name</i>	<i>Description</i>
<i>1</i>	<i>Ground</i>	<i>Ground wire connected to the ground of system</i>
<i>2</i>	<i>Vin</i>	<i>Powers the motor typically +5V is used</i>
<i>3</i>	<i>PWM digital pin</i>	<i>PWM signal is given in through this wire to drive the motor</i>

Table 2.6

2.5.3 Characteristics and features of servo motor

- Operating voltage is usually + 5V
- Torque: 2.5 kg / cm
- The operating speed is 0.1s / 60 °
- Rotation: 0 ° -180 °
- Engine weight: 9gm

2.6 Ultrasound Sensor

2.6.1 Overview

Ultrasound sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. by recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sensor and the object.



Figure 2.9

2.6.2 Effectiveness of Ultrasound

It detects the distance of the closest object in front of the sensor (from 3 cm up to 400 cm) , sound travels through air at about 344 m/s we have divided the product of speed and time by 2 because the time is the total time it took to reach the obstacle and return back .

$$\text{Distance} = (\text{Time} \times \text{Speed of Sound}) / 2$$

2.6.3 Timing Diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion.

We can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula:

$uS / 58 = \text{centimeters}$ or $uS / 148 = \text{inch}$; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal[8].

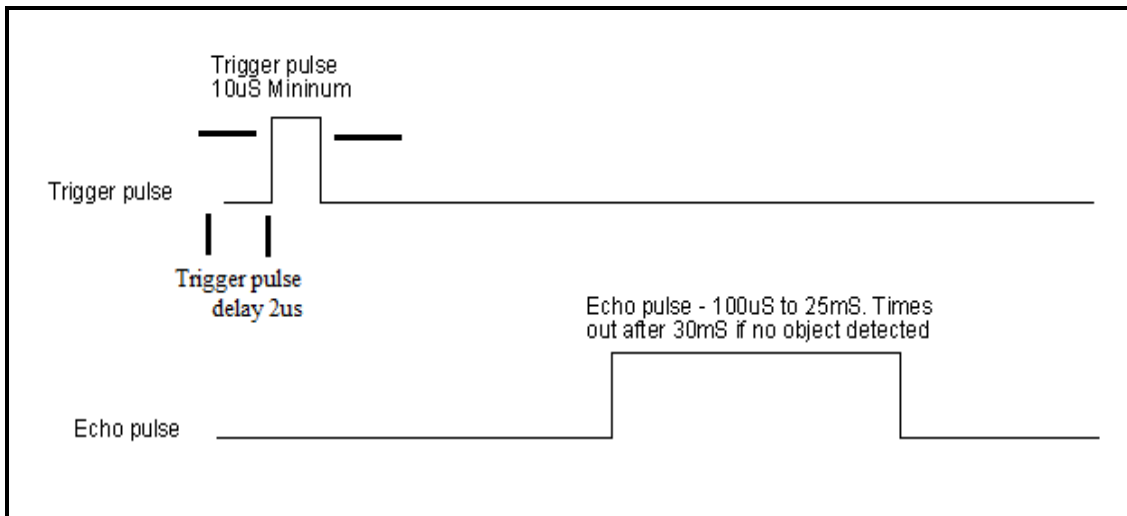


Figure 2.10

2.6.3 Plugging in the sensor

The Trig pin will be used to send the signal and the Echo pin will be used to listen for returning signal.

Pin	Type	Description
1	Trig	Digital Output Signal
2	Echo	Digital Input Signal
3	VCC	Power 5v
4	GND	Power ground

Table 2.7

2.7 FPV Camera

2.7.1 Overview

The FPV Camera is used to transfer the image of the place where the robot is located via Wi-Fi network using an application on the Smartphone, and the image is transferred and viewed over the phone by installing the phone on the base on the remote control or using the 3D glasses and changing the video mode to 3D through the application On your Smartphone.

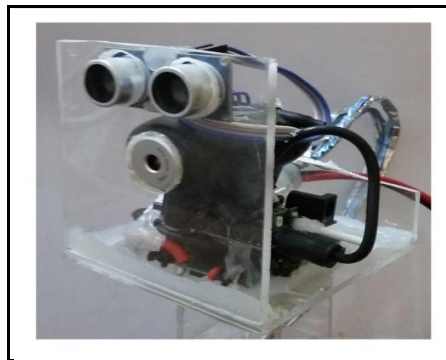


Figure 2.11

2.7.2 Promark VR Application

A app which we can control a wifi camera module , it's also display a real time video taken by the wifi camera module, which include support take photo and record video and 3D view.



Figure 2.12

2.8 Battery

2.8.1 Lithium Battery for microcontroller Unit

The Microcontroller Unit runs through an external Battery, the battery capacity (Amp-Hour) is determined by the total currents consumed through the attached devices on the robot [9].



Figure 2.13

2.8.2 Table of Loads

<i>Num</i>	<i>Device</i>	<i>Current</i>
1	Arduino Mega	50 mA
2	Ultrasound Sensor	15 mA
3	Camera	70 mA
4	nRF	12mA
5	Servo motor*2	100mA
<i>Total = 247mA</i>		

Table 2.8

We can calculate the capacity of battery (C) by:

$$C = \text{Amps} * \text{Time}$$

We chose lithium battery 7.2v with capacity 1500 mAh, so the battery can supply the robot about 6 hour .

$$\text{Time} = C / \text{Amps}$$

2.8.3 Specifications

<i>Voltage: 7V</i>	<i>Nominal capacity (9mA, off 5.4V) 1500mAh</i>
<i>Capacity: 1500 mAh</i>	<i>Operating temperature -40 ~ +85 ° C</i>
<i>Max. constant current 120Ma</i>	<i>Weight: 70g</i>

Table 2.9

2.8.4 Lithium Battery for H-Bridge

The robot runs through an external Battery, the battery capacity (Amp-Hour) is determined by the total currents consumed through the attached devices on the robot.



Figure 2.14

We can calculate the capacity of battery (C) by:

We chose lithium battery 12v with capacity 1500 mAh, so the battery can supply the robot about 10hour .

$$Time = C / Amps$$

2.8.6 Specifications

<i>Voltage: 12V</i>	<i>Nominal capacity (4800mA, off 6) 5000mAh</i>
<i>Capacity: 5000 mAh</i>	<i>Operating temperature -40 ~ +85 ° C</i>
<i>Max. constant current 120Ma</i>	<i>Weight: 70g</i>

Table 2.8

2.9 Robot Body

The body is designed to conform to the general shape of the robot and to prepare the robot's outer shape to give a consistent geometric design.



Figure 2.15

The Wheels are attached to a serrated tape that allows the robot to traverse the obstacles and rough surfaces in a flexible way and helps rotate and control the direction.



Figure 2.16

2.10 Circuit Connecting Devices with microcontroller

2.10.1 Circuit Diagram

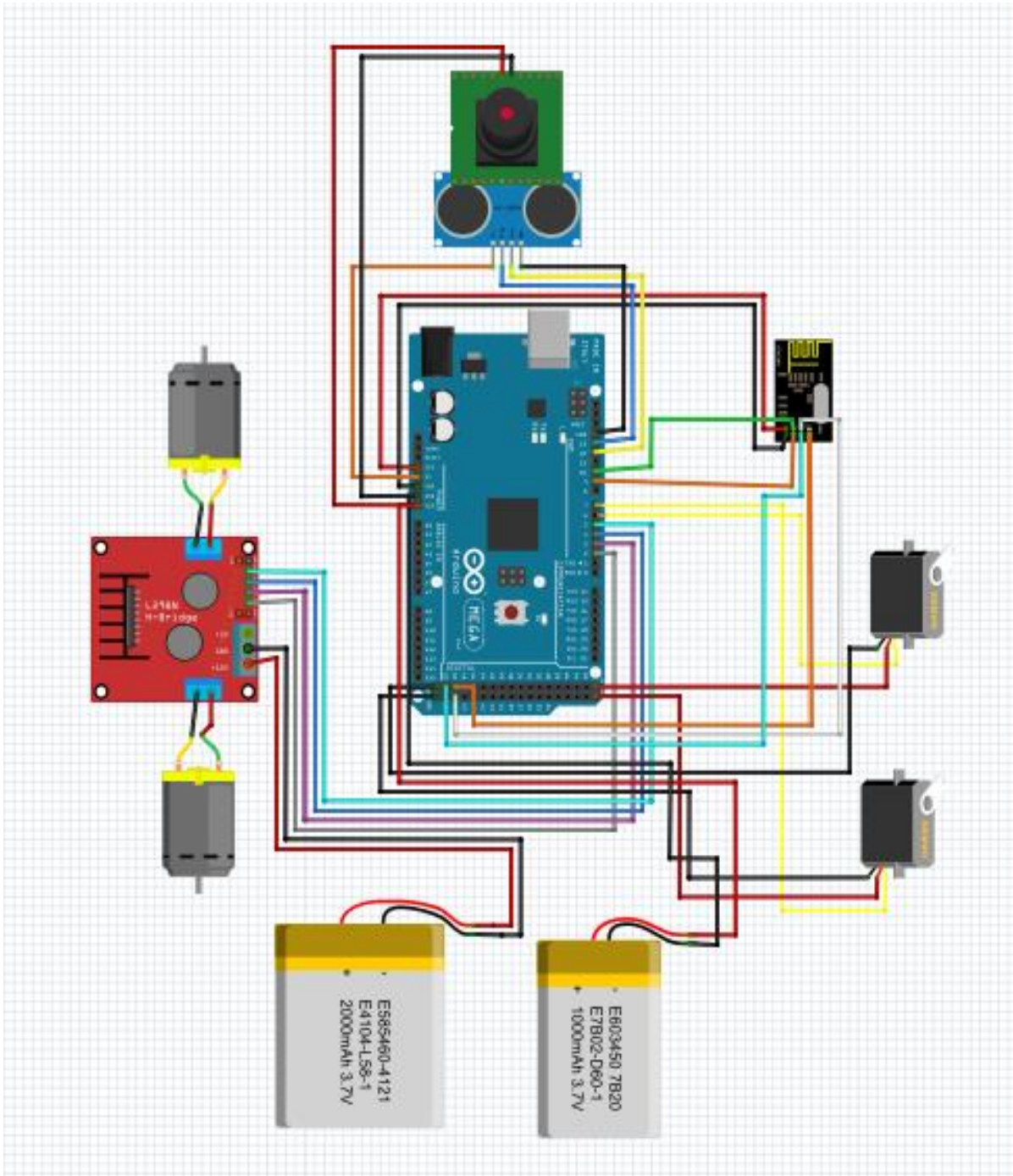


Figure 2.17

2.10.2 Electrical Circuit

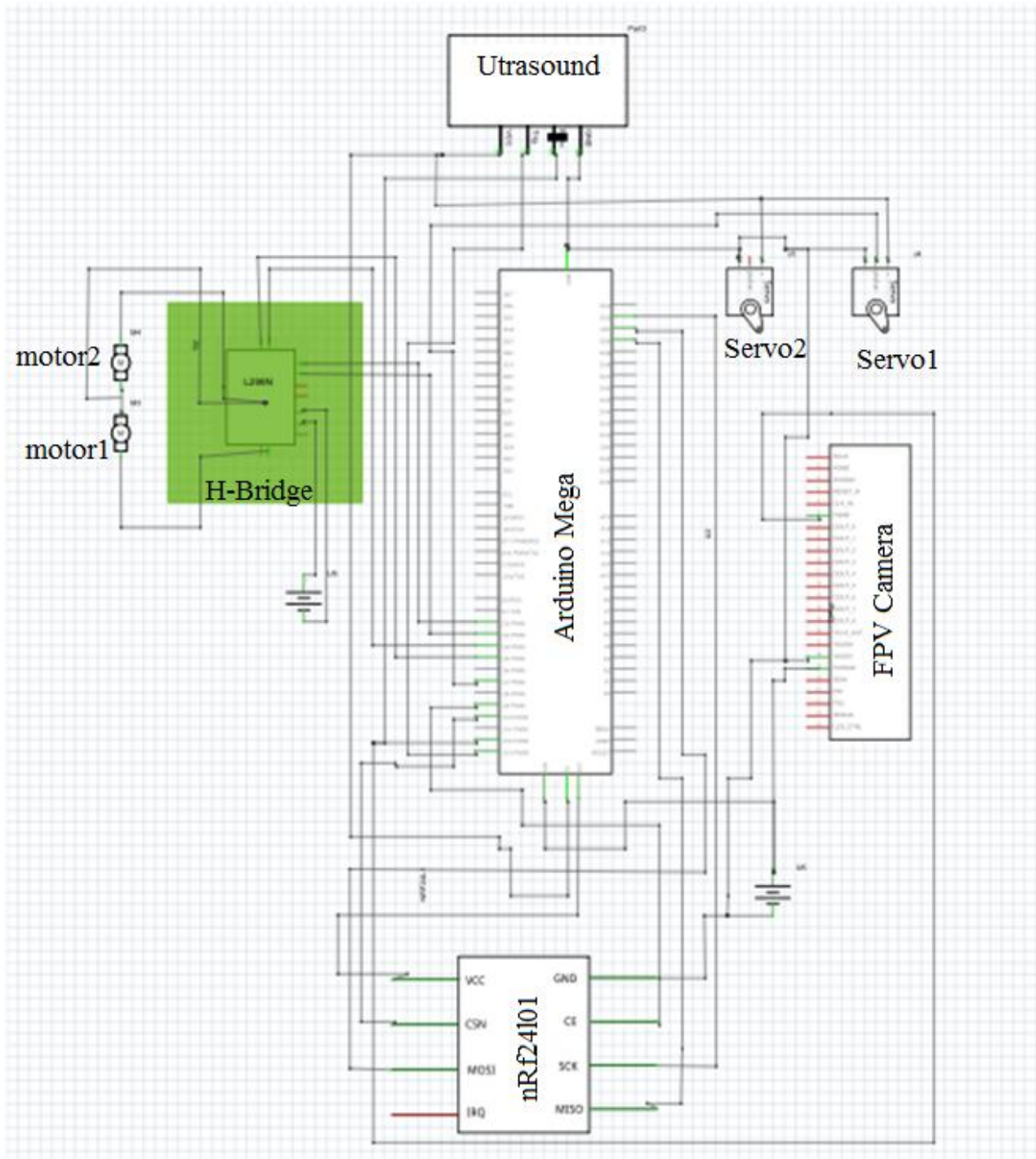
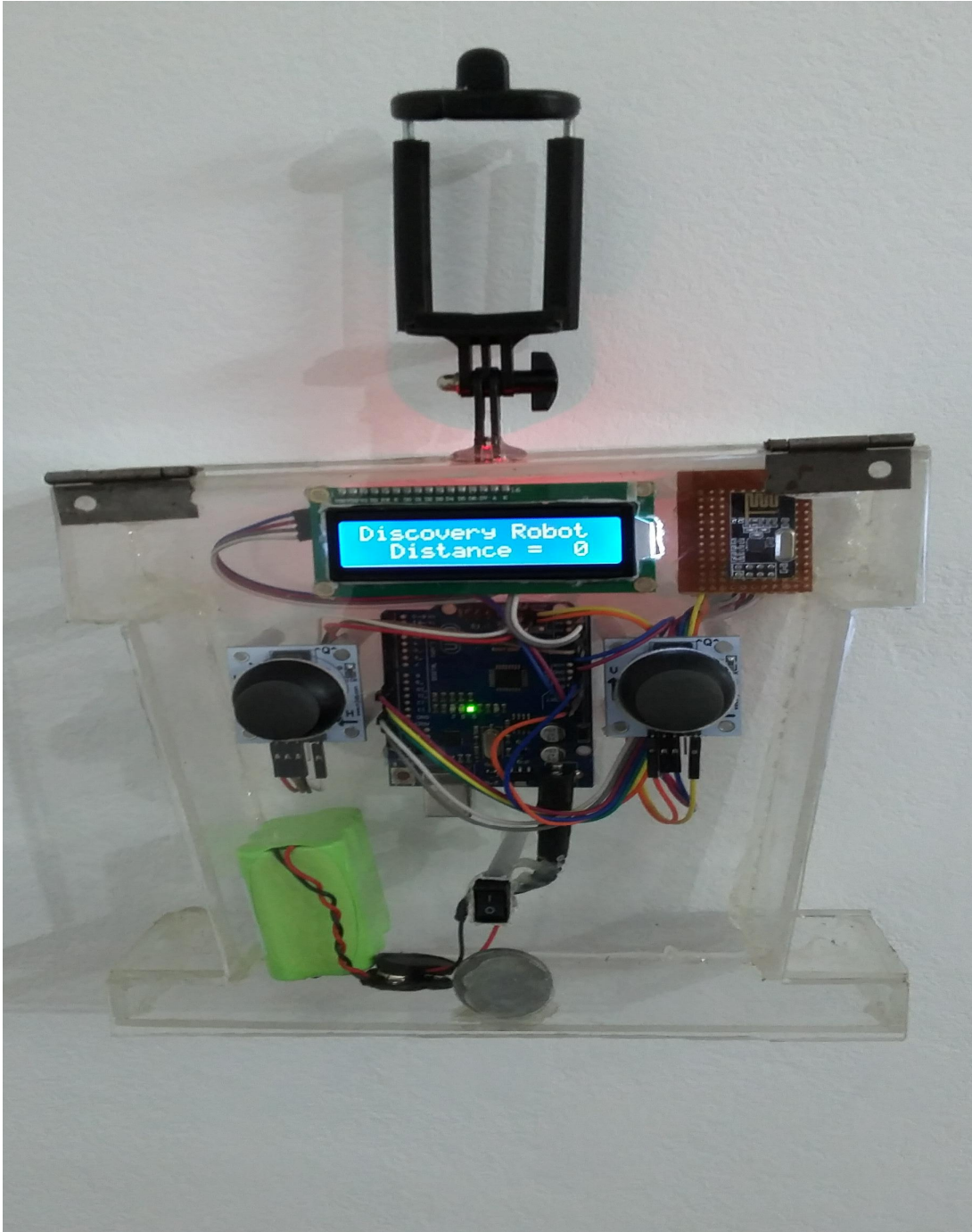


Figure 2.18

Chapter three

Remote Control Module



3.1 Microcontroller (Arduino Uno)

3.1.1 Overview

Is a microcontroller board 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, and a reset button. It contains everything needed to support the microcontroller, board can be powered via the external power supply using battery[3].

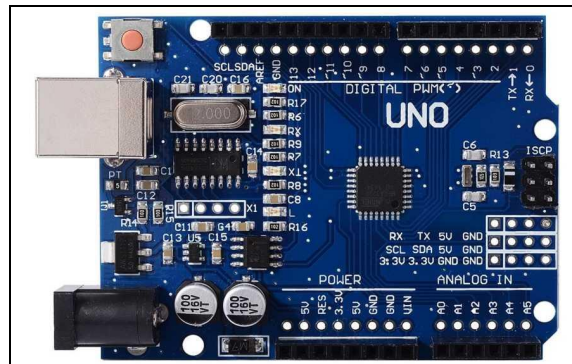


Figure3.1

3.1.2 Power

Power The Arduino Uno board can be powered via the external power supply using battery. Leads from a battery can be inserted in the GND and VIN pin headers of the POWER connector. The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board.

The recommended range is 7 to 12 volts. The power pins are as follows:

- VIN: The input voltage to the Arduino/Uno board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND: Ground pins.

3.1.3 Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pin Mode (), digital Write (), and digital Read () functions. They operate at 5 volts...PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analog Write () function.SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference () function.AREF. Reference voltage for the analog inputs. Used with analog Reference ().

3.1.4 Memory

The Arduino Uno has 32 KB (with 0.5 KB occupied by the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

3.1.5 Technical Specifications

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

Table 3.1

3.2 Wireless module

3.2.1 Overview

A wireless module that sends and receives commands in a wireless way and is connected to different Arduino types to execute incoming and future orders from different controllers.

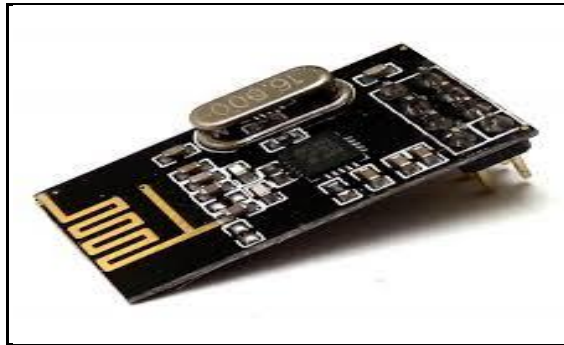


Figure 3.2

3.2.2 Wireless Module nRF24L01

Modem of NRF works on 2.4GHz frequency and has 100m range (at best). You can communicate with different baud rates: 250 kbps, 1 or 2 Mbps. High baud rate allows users even to send video material in 720p resolution. During receive power consumption is around 12mA and during transmission 11mA. NRF can operate on voltages between 1.9 and 3.6V. The input pins tolerate the 5V logic, so without any problems this module can be connected to the most popular development boards like Arduino. Most models of Arduino (Uno, Mega, Nano and more) are working with this logic levels[4].

3.2.3 Signal Range

The signal range of nRF24L01 Module if used in open space and with lower baud rate its range can reach up to 100 meter.

3.2.4 Network Configuration

An nRF24L01 configured as primary RX (PRX) will be able to receive data through 6 different data pipes that have a unique address but share the same frequency channel. This means that up to 6 different nRF24L01 configured as primary TX (PTX) can communicate with one nRF24L01 configured as PRX, and the nRF24L01 configured as PRX will be able to distinguish between them. Data pipe 0 has a unique 40-bit configurable address.

Each of data pipes 1-5 has an 8-bit unique address and shares the 32 most significant address bits. All data pipes can perform full Enhanced functionality. nRF24L01 will use the data pipe address when acknowledging a received packet. This means that nRF24L01 will transmit with the same address as it receives payload at. In the PTX device, data pipe 0 is used to receive the acknowledge, and therefore the receive address for data pipe 0 has to be equal to the transmit address to be able to receive the acknowledge.

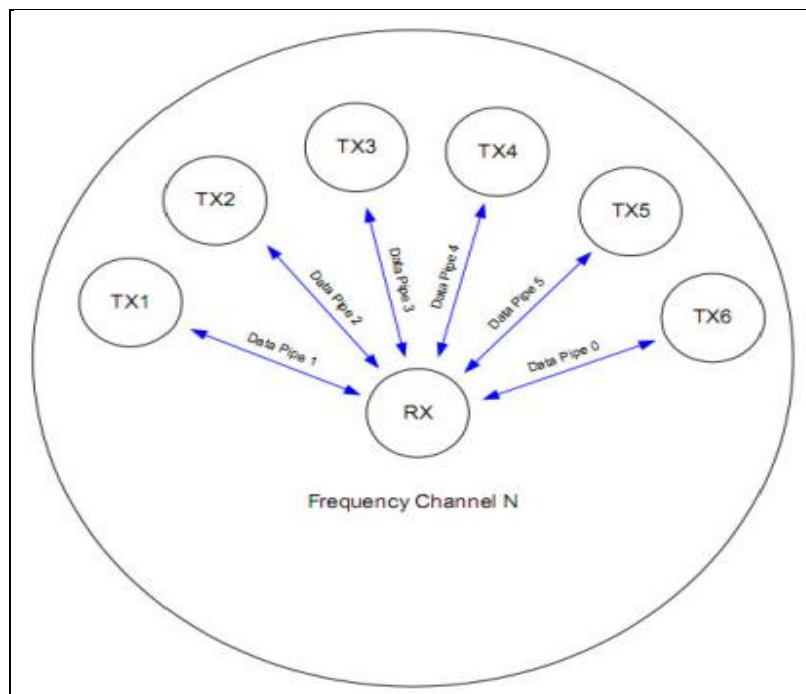


figure3.3

3.2.5 Transmitting Payload (TX)

- The configuration bit PRIM_RX has to be low, address settings have to be performed for the TX device and the RX device:

TX device: TX_ADDR = 0xB3B4B5B605

RX device: RX_ADDR = 0xB3B4B5B605

- A high pulse on CE starts the transmission. The minimum pulse width on CE is 10 μ s.
- nRF24L01 Shock Burst:

* Radio is powered up

*16 MHz internal clock is started.

*RF packet is completed (see the packet description)

* Data is transmitted at high speed (1 Mbps or 2 Mbps configured by MCU).

- The IRQ pin will be active when MAX_RT or TX_DS is high. To turn off the IRQ pin, the interrupt source must be reset by writing to the status register (see Interrupt chapter). If no acknowledgement is received for a packet after the maximum number of retries, no further packets can be sent before the MAX_RX interrupt is cleared.
- The device goes into Standby-I mode if CE is low. Otherwise next payload in TXFIFO will be sent. If TX FIFO is empty and CE is still high, the device will enter Standby -II mode.
- If the device is in Standby-II mode, it will go to Standby-I mode immediately if CE is set low.

3.2.6 Receive Payload (RX)

- RX is selected by setting the PRIM_RX bit in the configuration register to high, all data pipes that shall receive data must be enabled (EN_RXADDR register), auto acknowledgement for all pipes running Enhanced ShockBurst has to be enabled (EN_AA register), and the correct payload widths must be set (RX_PW_Px registers).

- Addresses have to be set up as described in item 2 in the Enhanced ShockBurst transmit payload chapter above.
- Active RX mode is started by setting CE high.
- After 130µs nRF24L01 is monitoring the air for incoming communication.
- When a valid packet has been received (matching address and correct CRC), the payload is stored in the RX-FIFO, and the RX_DR bit in status register is set high. The IRQ pin will be active when RX_DR is high. RX_P_NO in status register will indicate what data pipe the payload has been received in.
- If auto acknowledgement is enabled, an acknowledgement is sent back.
- MCU sets the CE pin low to enter Standby-I mode (low current mode).
- MCU can clock out the payload data at a suitable rate via the SPI interface.
- The device is now ready for entering TX or RX mode or power down mode.

3.2.7 Pin Functions

<i>Pin</i>	<i>Name</i>	<i>Description</i>	<i>Pin function</i>	<i>Description</i>
1	CE		Digital Input	Chip Enable Activates RX or TX mode
2	CSN		Digital Input	SPI Chip Select
3	SCK		Digital Input	SPI Clock
4	MOSI		Digital Input	SPI Slave Data Input
5	MISO		Digital Input	SPI Slave Data Output, with tri-state option
6	IRQ		Digital Input	Maskable interrupt pin
7	VDD		Power	Power Supply (+3V DC)
8	VSS		Power	Ground (0V)

Table 3.2

3.3 Joysticks

3.3.1 Overview

is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling. It is used to control the dc motor and motor driver to control the direction of the robot and control the servo motor to control the direction of the camera's rotation. This type uses 2-axis (x, y) which means it could record movement in 2 directions: top to bottom and left to right, or horizontal and vertical movement.



figure3.4

3.3.2 Description Joystick

This module produces an output of around 2.5V from X and Y when it is in its resting position. Moving the joystick will cause the output to vary from 0V to 5V depending on its direction. If you connect this module to a microcontroller, you can expect to read a value of around 512 in its resting position (expect small variations due to tiny imprecision of the springs and mechanism). When you move the joystick, you should see the values change from 0 to 5 volts depending on its position, and the analog input in Arduino change this value from 0 to 1024 by scaling function (mapping)[8].

3.3.3 Joystick control

The joystick on the right controls the movement of the robot in four directions (forward, backward, right, left), The readings are taken from the sticks through variable resistance, varying from (0 to 5) volts, and sending the value to the microcontroller to control the direction of the robot And its speed.

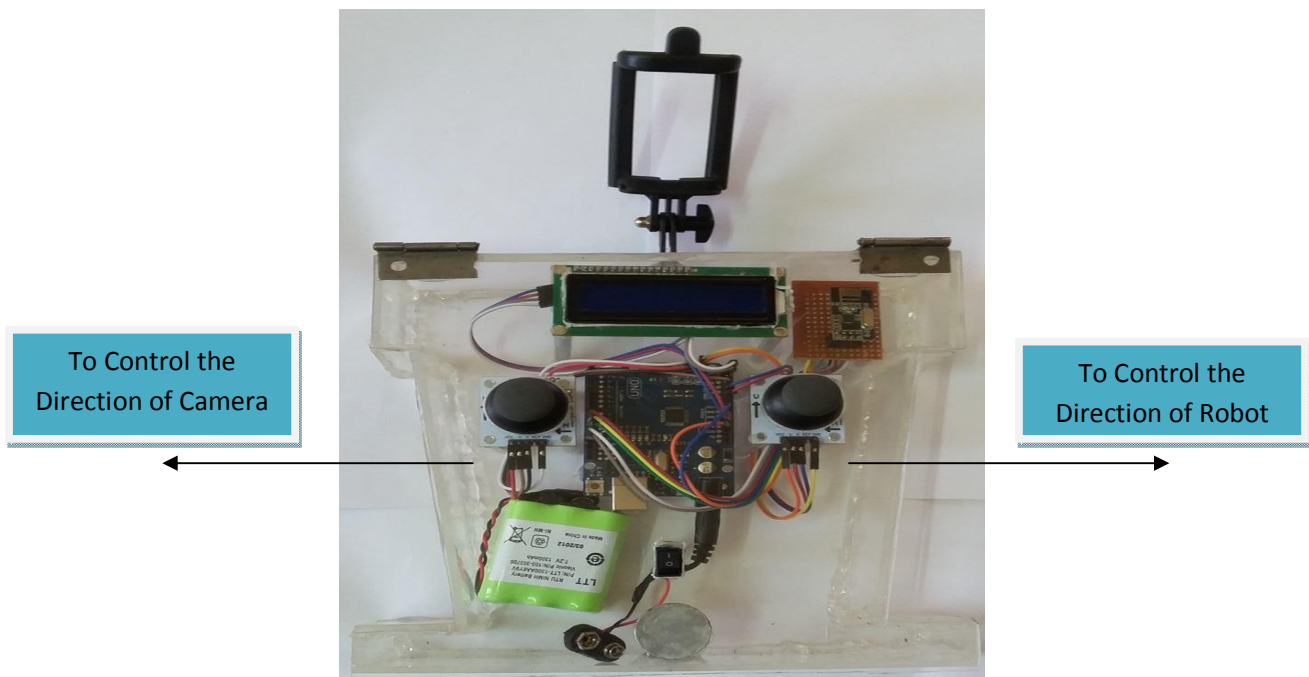


Figure 3.5

The stick on the left controls the movement of the camera in the four directions (up, down, right, left). The readings are taken from the sticks through variable resistance to the microcontroller to control the direction of the camera.

3.3.4 Pin Functions

Pin	Name	Description	Pin function	Description
1	V		Analog Input	Control Vertical Direction
2	H		Analog Input	Control Horizontal Direction
3	VCC		Power	Power Supply (+5V DC)
4	GND		Power	Ground (0V)

Table 3.3

3.4 Liquid Crystal Display (LCD)

3.4.1 Overview

LCD (Liquid Crystal Display) screen is an electronic display module for a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs.



Figure 3.6

3.4.2 LCD LCM1602 IIC

This is display screen with I2C interface. It is able to display 16x2 characters on 2 lines, white characters on blue background, Arduino LCD display projects will run out of pin resources easily, especially with Arduino Uno. And it is also very complicated with the wire soldering and connection, this I2C 16x2 Arduino LCD Screen is using an I2C communication interface, it means it only needs 4 pins for the LCD display: VCC, GND, SDA, SCL. It will saves at least 4 digital / analog pins on Arduino, all connector are standard XH2.54 (Breadboard type), we can connect with jumper wire directly, and we can write on the screen easily through a special library in Arduino programming[8].

3.4.3 LCM Specification

- Compatible with Arduino UNO, Mega, Nano, Mini
- I2C Address:0x20-0x27(0x20 default)
- Back lit (Blue with white char color)
- Supply voltage: 5V
- Interface:I2C/TWI x1,Gadgeteer interface x2
- Adjustable contrast (MANUAL)
- Size:82x35x18 mm (3.2x1.4x0.7 in)

3.4.4 Pin Functions

<i>Pin</i>	<i>Name Description</i>	<i>Pin function</i>	<i>Description</i>
1	SDA	Analog Input	I2C Data Line
2	SCL	Analog Input	I2C Clock
3	VCC	Power	Power Supply (+5V DC)
4	GND	Power	Ground (0V)

Table3.4

3.5 Lithium Battery

3.5.1 Overview

The Remote runs through an external Battery, the battery capacity (Amp-Hour) is determined by the total currents consumed through the attached devices on the remote[9].



Figure 3.7

3.5.2 Table of Loads

<i>Num</i>	<i>Device</i>	<i>Current</i>
1	Arduino UNO	50 mA
2	NRF24L1	12 mA
3	Joystick	40mA
4	LCD	30 mA
<i>Total = 132 mA</i>		

Table 3.5

We can calculate the capacity of battery (C) by:

$$C = \text{Amps} * \text{Time}$$

We chose lithium battery 7v with capacity 1300 mAh, so the battery can supply the remote about 9 hour .

$$\text{Time} = C / \text{Amps}$$

3.5.3 Specifications of Battery

<i>Voltage: 7.2V</i>	<i>Nominal capacity 1300mAh</i>
<i>Capacity: 1200 mAh</i>	<i>Operating temperature -40 ~ +85 ° C</i>
<i>Max. constant current 150mA</i>	<i>Weight: 90.5g</i>

Table 3.6

3.6 Remote Body

The remote control body is designed from the reinforced plastic and it is designed to be containing the all remote parts.



Figure 3.8

And it was attached to a phone-carrying base to show what was around us to easier to control and move the robot.



Figure 3.9

3.7 Circuit Connecting Devices with microcontroller

3.7.1 Circuit Diagram

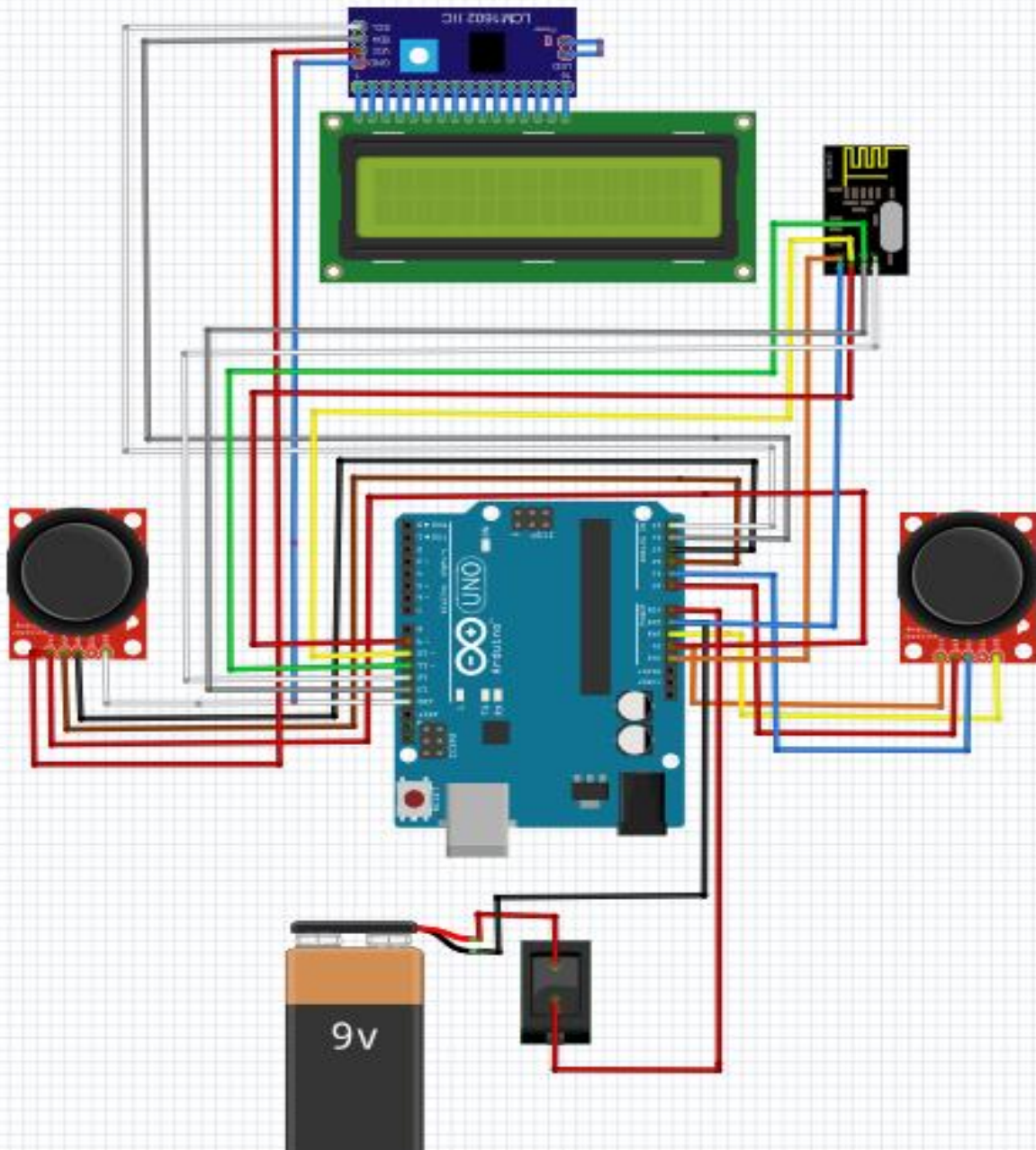


Figure 3.10

3.7.2 Electrical Circuit

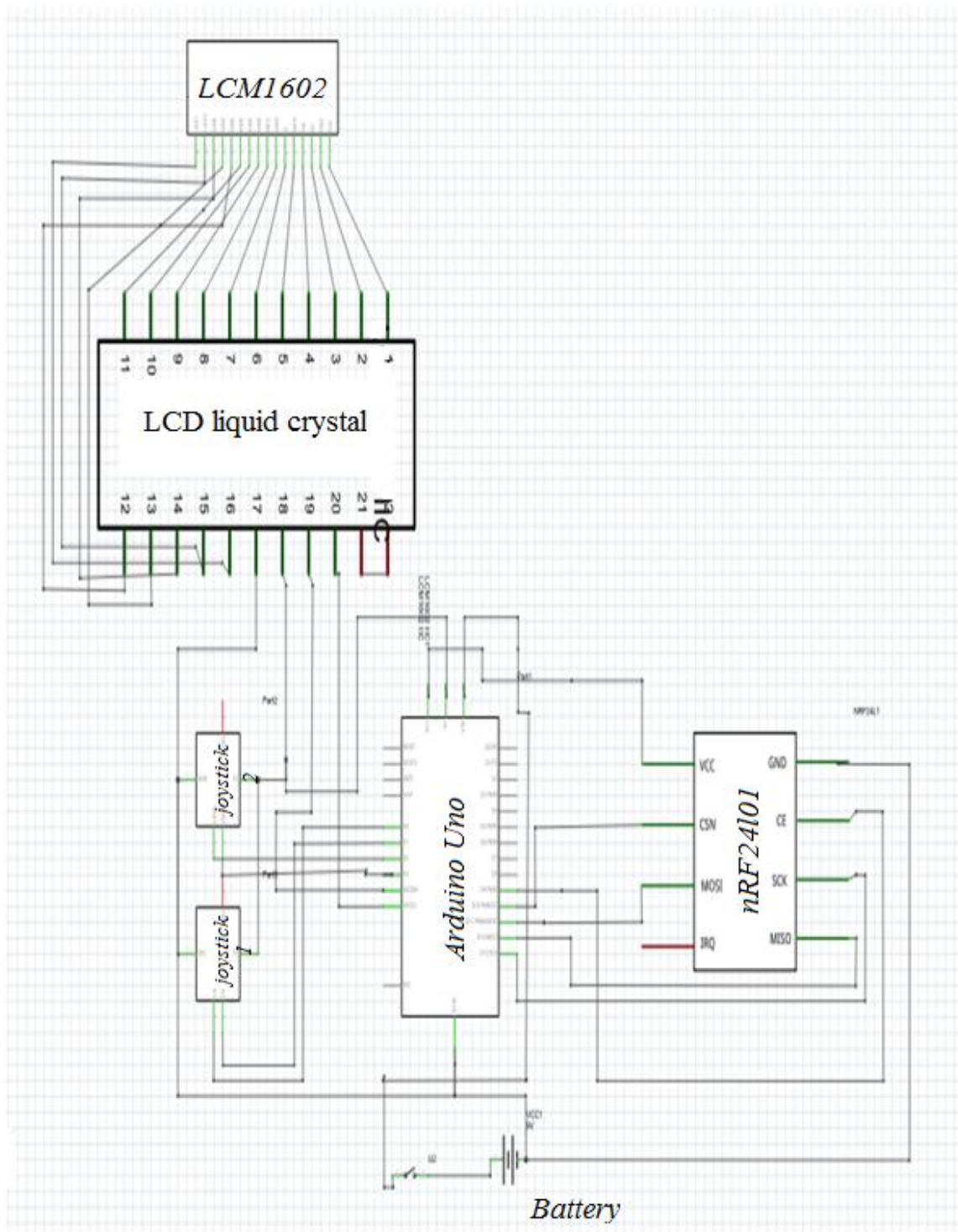


Figure 3.11

Chapter Four

3D Glasses Module



4.1 Microcontroller (Arduino Nano)

4.1.1 Overview

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P . It has more or less the same functionality of the other Arduino , but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one[3].

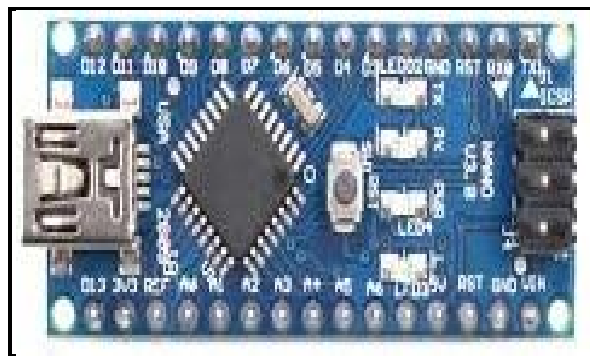


Figure4.1

4.1.2 Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

4.1.3 Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using `pinMode ()`, `digitalWrite ()`, and `digitalRead ()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. 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.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the `analogWrite ()` function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is high value, the LED is on, when the pin is LOW, it's off.

The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, although it is possible to change the upper end of their range using the `analogReference ()` function. Analog pins 6 and 7 can not be used as digital pins. Additionally, some pins have specialized functionality:

I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the `Wire` library (documentation on the Wiring website).

4.1.4 Memory

The Arduino Nano has 32 KB, (also with 2 KB used for the bootloader). The Nano has 2 KB of SRAM and 1 KB of EEPROM.

4.1.5 Technical Specifications

Microcontroller	ATmega328
<i>Operating Voltage</i>	<i>5 V</i>
<i>Flash Memory</i>	<i>32 KB of which 2 KB used by bootloader</i>
<i>SRAM</i>	<i>2 KB</i>
<i>Clock Speed</i>	<i>16 MHz</i>
<i>Analog IN Pins</i>	<i>8</i>
<i>EEPROM</i>	<i>1 KB</i>
<i>DC Current per I/O Pins</i>	<i>40 mA (I/O Pins)</i>
<i>Input Voltage</i>	<i>7-12 V</i>
<i>Digital I/O Pins</i>	<i>22 (6 of which are PWM)</i>
<i>PWM Output</i>	<i>6</i>
<i>Power Consumption</i>	<i>19 mA</i>
<i>PCB Size</i>	<i>18 x 45 mm</i>
<i>Weight</i>	<i>7 g</i>

Table 4.1

4.2 Wireless Module

4.2.1 Overview

A wireless module that sends and receives commands in a wireless way and is connected to different Arduino types to execute incoming and future orders from different controllers.

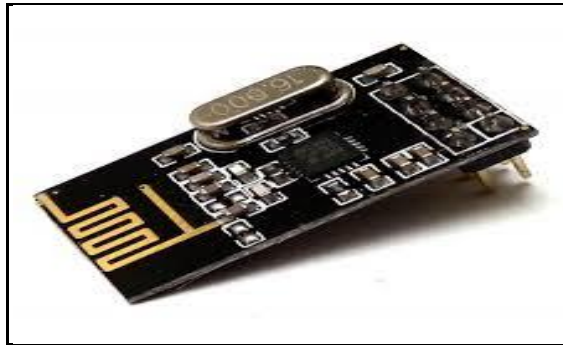


Figure 4.2

4.2.2 Wireless module nRF24L01

Modem of NRF works on 2.4GHz frequency and has 100m range (at best). You can communicate with different baud rates: 250 kbps, 1 or 2 Mbps. High baud rate allows users even to send video material in 720p resolution. During receive power consumption is around 12mA and during transmission 11mA. NRF can operate on voltages between 1.9 and 3.6V. The input pins tolerate the 5V logic, so without any problems this module can be connected to the most popular development boards like Arduino. Most models of Arduino (Uno, Mega, Nano and more) are working with this logic levels[4].

4.2.3 Signal Range

The signal range of nRF24L01 Module if used in open space and with lower baud rate its range can reach up to 100 meter.

4.2.4 Network Configuration

An nRF24L01 configured as primary RX (PRX) will be able to receive data through 6 different data pipes that have a unique address but share the same frequency channel. This means that up to 6 different nRF24L01 configured as primary TX (PTX) can communicate with one nRF24L01 configured as PRX, and the nRF24L01 configured as PRX will be able to distinguish between them. Data pipe 0 has a unique 40-bit configurable address.

Each of data pipes 1-5 has an 8-bit unique address and shares the 32 most significant address bits. All data pipes can perform full Enhanced functionality. nRF24L01 will use the data pipe address when acknowledging a received packet. This means that nRF24L01 will transmit with the same address as it receives payload at. In the PTX device, data pipe 0 is used to receive the acknowledge, and therefore the receive address for data pipe 0 has to be equal to the transmit address to be able to receive the acknowledge.

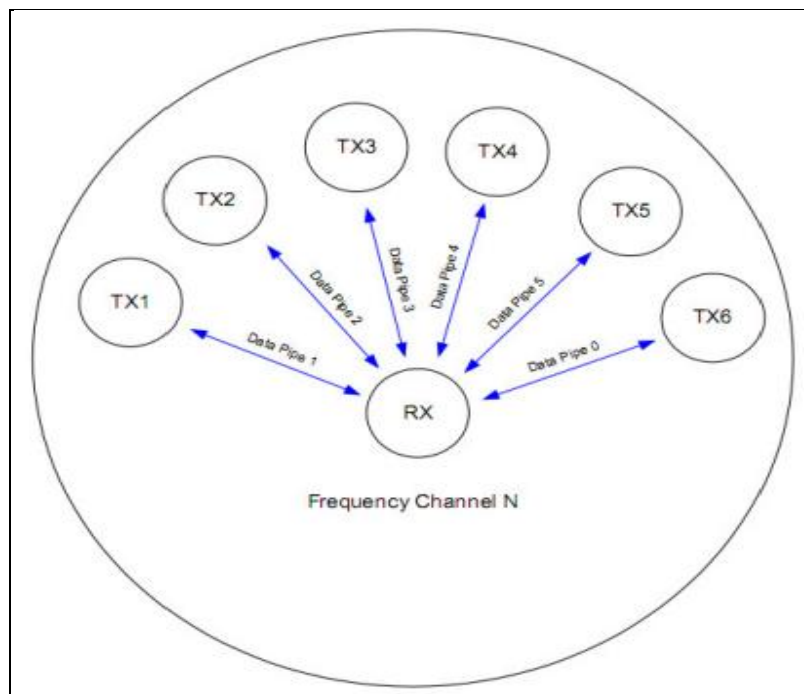


Figure4.3

4.2.5 Transmitting Payload (TX)

- The configuration bit PRIM_RX has to be low, address settings have to be performed for the TX device and the RX device:
TX device: TX_ADDR = 0xB3B4B5B605
RX device: RX_ADDR = 0xB3B4B5B605
- A high pulse on CE starts the transmission. The minimum pulse width on CE is 10 μ s.
- nRF24L01 Shock Burst:
 - Radio is powered up
 - *16 MHz internal clock is started.
 - RF packet is completed (see the packet description)
 - Data is transmitted at high speed (1 Mbps or 2 Mbps configured by MCU).
- The IRQ pin will be active when MAX_RT or TX_DS is high. To turn off the IRQ pin, the interrupt source must be reset by writing to the status register (see Interrupt chapter). If no acknowledgement is received for a packet after the maximum number of retries, no further packets can be sent before the MAX_RX interrupt is cleared.
- The device goes into Standby-I mode if CE is low. Otherwise next payload in TXFIFO will be sent. If TX FIFO is empty and CE is still high, the device will enter Standby -II mode.
- If the device is in Standby-II mode, it will go to Standby-I mode immediately if CE is set low.

4.2.6 Receive Payload (RX)

- RX is selected by setting the PRIM_RX bit in the configuration register to high, all data pipes that shall receive data must be enabled (EN_RXADDR register), auto acknowledgement for all pipes running

Enhanced ShockBurst has to be enabled (EN_AA register), and the correct payload widths must be set (RX_PW_Px registers).

- Addresses have to be set up as described in item 2 in the Enhanced ShockBurst transmit payload chapter above.
- Active RX mode is started by setting CE high.
- After 130µs nRF24L01 is monitoring the air for incoming communication.
- When a valid packet has been received (matching address and correct CRC), the payload is stored in the RX-FIFO, and the RX_DR bit in status register is set high. The IRQ pin will be active when RX_DR is high. RX_P_NO in status register will indicate what data pipe the payload has been received in.
- If auto acknowledgement is enabled, an acknowledgement is sent back.
- MCU sets the CE pin low to enter Standby-I mode (low current mode).
- MCU can clock out the payload data at a suitable rate via the SPI interface.
- The device is now ready for entering TX or RX mode or power down mode.

4.2.7 Pin Functions

<i>Pin</i>	<i>Name</i>	<i>Description</i>	<i>Pin function</i>	<i>Description</i>
1	CE		Digital Input	Chip Enable Activates RX or TX mode
2	CSN		Digital Input	SPI Chip Select
3	SCK		Digital Input	SPI Clock
4	MOSI		Digital Input	SPI Slave Data Input
5	MISO		Digital Input	SPI Slave Data Output, with tri-state option
6	IRQ		Digital Input	Maskable interrupt pin
7	VDD		Power	Power Supply (+3V DC)
8	VSS		Power	Ground (0V)

Table 4.2

4.3 MPU Sensor

4.3.1 Overview

The MPU sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino[10].

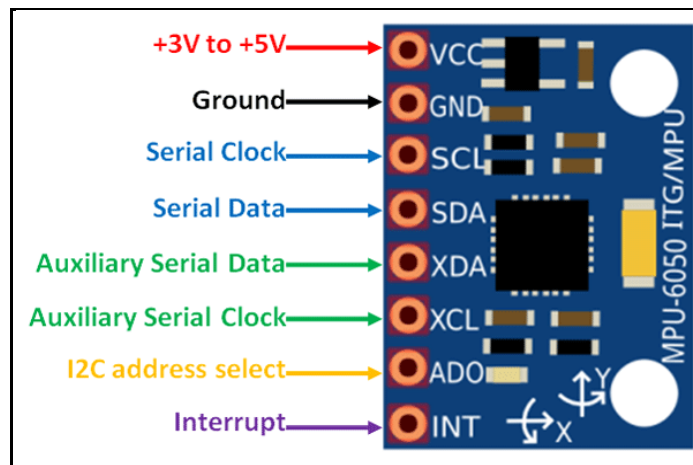


Figure 4.4

4.3.2 MPU-6050

The MPU-6050 sensor used to take the coordinates of the 3D glasses and adjusts the camera according to the coordinates of the glasses, to see the area to be explored and it always acts as a slave to the Arduino with the SDA and SCL pins connected to the I2C-bus.

4.3.3 Pin Configuration

Pin	Pin Name	Description
1	Vcc	Provides power for the module, can be +3V to +5V. Typically +5V is used
2	Ground	Connected to Ground of system
3	Serial Clock (SCL)	Used for providing clock pulse for I2C Communication
4	Serial Data (SDA)	Used for transferring Data through I2C communication
5	Auxiliary Serial Data (XDA)	Can be used to interface other I2C modules with MPU6050. It is optional
6	Auxiliary Serial Clock (XCL)	Can be used to interface other I2C modules with MPU6050. It is optional
7	AD0	If more than one MPU6050 is used a single MCU, then this pin can be used to vary the address
8	Interrupt (INT)	Interrupt pin to indicate that data is available for MCU to read.

Table 4.3

4.4 Battery

4.4.1 Overview

The 3D Glasses runs through an external Battery, the battery capacity (Amp-Hour) is determined by the total currents consumed through the attached devices on the glasses[9].



Figure 4.5

4.4.2 Table of Loads

<i>Num</i>	<i>Device</i>	<i>Current</i>
1	Arduino Nano	50 mA
2	nRF24L01	12 mA
3	MPU	40 mA
<i>Total = 102mA</i>		

Table 4.4

We can calculate the capacity of battery (C) by:

$$C = \text{Amps} * \text{Time}$$

We chose lithium battery 9v with capacity 12000 mAh, so the battery can supply the 3D Glasses about 11 hour.

$$\text{Time} = C / \text{Amps}$$

4.4.3 Specifications

<i>Voltage: 9V</i>	<i>Nominal capacity (9mA, off 5.4V) 1200mAh</i>
<i>Capacity: 1200 mAh</i>	<i>Operating temperature -40 ~ +85 ° C</i>
<i>Max. constant current 120mA</i>	<i>Weight: 50.5g</i>

Table 4.5

4.5 VR Glasses

4.5.1 Overview

A Virtual Reality (VR) glass are one of the most wearable technologies, and have become the most important accessories and accessories of smart phones and tablets, and is focused on the transfer of information to and from the processor that displays virtual reality. These glasses are made up of a piece that covers the eyes completely, and in front of each eye there is a lens - a small display screen - that displays images in 3D so that the eyes can take pictures from each lens separately, and then the brain will install the images to actually look three-dimensional.



Figure 4.6

4.5.2 Siswoo VR Glasses

Siswoo VR Glass Specification List		
VR Glass Detail	Visual Display Size	32inch
	Display Screen Size	165*85mm
	Product Material	PVC
	Available Phone Size	4inch -6inch phones
Package	Net Weight	290g
	Total Weight	554g
	Unit Package Dimension	220*165*125mm
Color	Black	

Table 4.6

4.6 Circuit Connecting Devices with microcontroller

4.6.1 Circuit Diagram

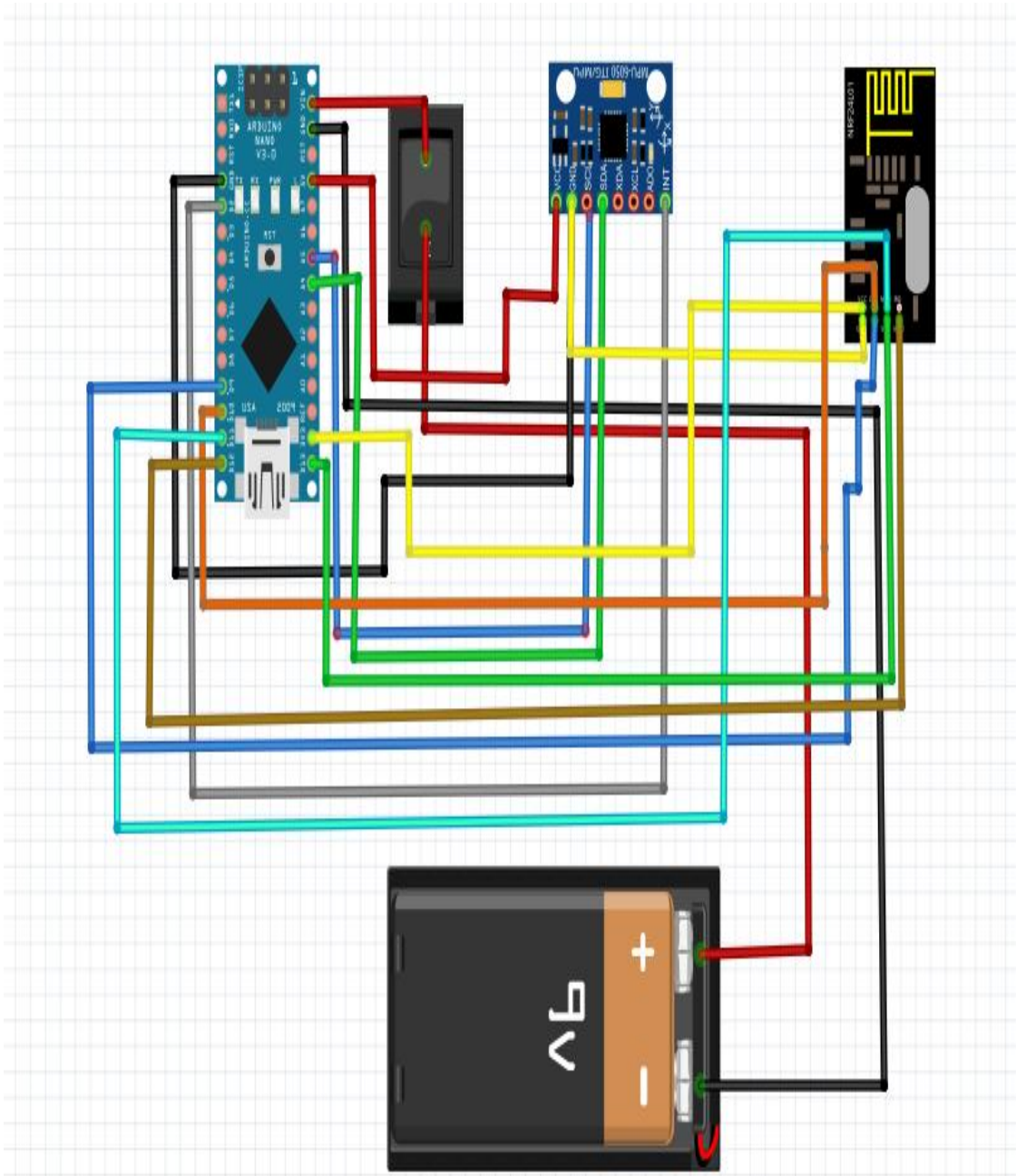


Figure 4.7

4.6.2 Electrical Circuit

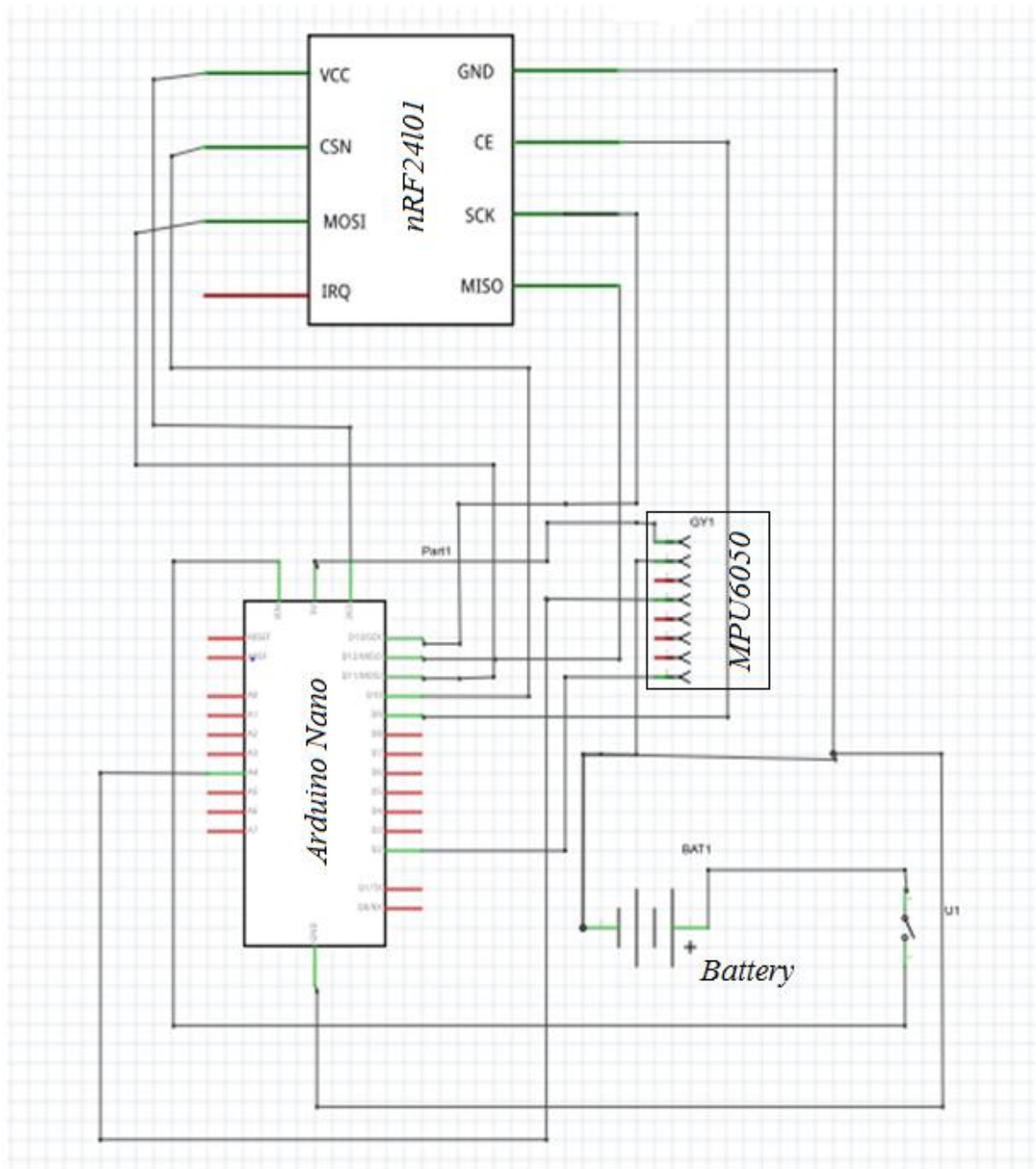


Figure 4.8

Chapter Five

Programming

5.1 Arduino Program

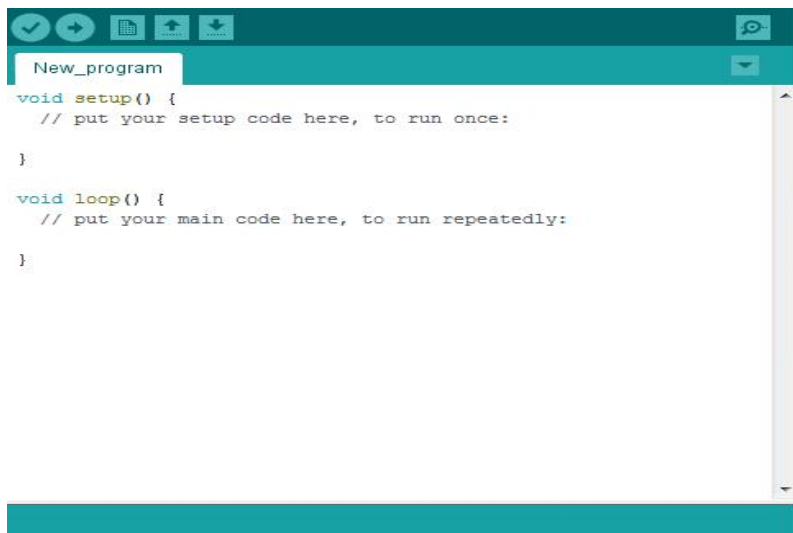
5.1.1 Overview

We chose Arduino boards for its able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, connect to many other actions Additionally, the Arduino code uses a simplified version of C++ making it easier to write program.

5.1.2 Structure

Structure The basic structure of the Arduino programming language is fairly simple and runs in at least two parts. These two required parts, or functions, enclose blocks of statements. Where setup" **Void setup ()**" is the preparation and "loop"" **Void loop ()**"is the execution. Both functions are required for the program to work. The setup function should follow the declaration of any variables at the very beginning of the program. It is the first function to run in the program, is run only once, and is used to set" pin Mode"or initialize serial communication, The loop function follows next and includes the code to be executed continuously-reading inputs, triggering outputs[3].

5.1.3 Sub-Program



```
void setup() {  
  // put your setup code here, to run once:  
  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

5.2 Robot Code

5.2.1 Code Description

The Robot communicates with the Remote Control and the 3D glasses through the wireless network, the data is received and transmitted across Array via wireless network and processed by the microcontroller through the code that is programmed to read the data such as the distance from the ultrasound sensor and send it to the display screen LCD, the data that receiving such as DC motor moving data and direction data for Servo Motor are processed and sent to the microcontroller output.

5.2.2 Robot Code

```
#include <SPI.h> //Call SPI library for communicate with the nRF24L01
#include <nRF24L01.h> //Call nRF2401 library
#include <RF24.h> // library which helps us to control the radio modem
#include <Servo.h> // Call Servo motor library
#define trigPin 13 // define pin 13 trigger pulse for ultrasound
#define echoPin 12 // define pin 12 echo pulse for ultrasound

const int pinCE = 9; //This pin is used to set the nRF24 to standby (0) or
active mode (1)
const int pinCSN = 10; //This pin is used to set the nRF24 SPI Chip
RF24 wirelessSPI (pinCE, pinCSN); //set pin number of CE and CSN for
NRF24l01 chip

const uint64_t readAddress [] = {0xB00B1E50D2LL, 0xB00B1E50C3LL};
// Address for reading pipe line in NRF
const uint64_t writeAddress[] = {0xB00B1E50B1LL, 0xB00B1E50A4LL};
// Address for writing pipe line in NRF
Servo myServo1; // define variable for servo motor
Servo myServo2; // define variable for servo motor
int mot2=2, mot3=3; // define motor1 pines
int mot4=4,mot5=5; // define motor2 pines
```

```

int recev[6]; // define array for storage the data in the receiving
int trans[6]; // define array for storage the data in the transmission

void setup () {
pinMode (trigPin, OUTPUT); // initialize the trigPin pin as an output
PinMode(mot2, OUTPUT); //initialize the motor1 pin1 as an output
PinMode (mot3, OUTPUT); //initialize the motor1 pin2 as an output
PinMode(mot4, OUTPUT); //initialize the motor2 pin1 as an output
PinMode(mot5, OUTPUT); //initialize the motor2 pin2 as an output
PinMode (echoPin, INPUT); // initialize the echoPin pin as an input

myServo1.attach(6); // initialize the myServo1 output for servo1
myServo2.attach(7); // initialize the myServo2 output for servo2
wirelessSPI.begin(); // turn on the wireless network
wirelessSPI.setPALevel(RF24_PA_MIN) // set the level for amplifier
power for modules are close to each other

wirelessSPI.openReadingPipe(1,readAddress[0]); // open pipe line for
reading from remote and select number pipe (1) and address (0)
wirelessSPI.openReadingPipe(2,readAddress[1]); // open pipe line for
reading from glasses and select number pipe (2) and address (1)
wirelessSPI.openWritingPipe(writeAddress[0]); // open pipe line for write
to remote and select number pipe (1) and address (0)
}
void loop()
{
wirelessSPI.startListening(); // start wireless-SPI listening to remote and
glasses
byte pipNum =0; // define variable to select num of pipe

while(wirelessSPI.available(&pipNum)) //while wireless start read or write
from pipe line for remote or glasses do the code
{wirelessSPI.read(&recev, sizeof(recev)); // start reading form remote and
glasses

```

```

if(pipNum==1) // if the pipe line read from remote perform if code
{ int val0=map(recev[0],0,1024,-250,250); //read the value from array (0)
then mapping the value from 0 - 1024 to -250 - 250
int val1=map(recev[1],0,1024,-250,250); //read the value from array (1)
then mapping the value from 0 - 1024 to -250 - 250
int val2=map(recev[2],0,1024,70,100); //read the value from array (2) then
mapping the value from 0 - 1024 to 70 – 100 degree
int val3=map(recev[3],0,1024,0,180); //read the value from array (3) then
mapping the value from 0 - 1024 to 0 – 180 degree

myServo1.write(val2); // Apply the value on the output of servo1
myServo2.write(val3); // Apply the value on the output of servo2
delay(5); // delay 5 millisecond

if (val0>10) // if the value grate than 10 perform if condition
{analogWrite(mot2,val0); // Apply the value(0) on the output of motor(1)
analogWrite(mot5,val0); // Apply the value(0) on the output of motor(2)
digitalWrite(mot3,LOW); //Apply low on the output of motor(1)
digitalWrite(mot4,LOW); //Apply low on the output of motor(2)
}
if (val0<-10) // if the value less than -10 perform if condition
{ val0=val0*-1; // change the value to positive
analogWrite(mot3,val0); // Apply the value(0) on the output of motor(1)
analogWrite(mot4,val0); // Apply the value(0) on the output of motor(2)
digitalWrite(mot2,LOW); //Apply low on the output of motor(1)
digitalWrite(mot5,LOW); //Apply low on the output of motor(2)

}
if (val1>10) // if the value grate than 10 perform if condition
{ analogWrite(mot2,val0); // Apply the value(1) on the output of motor(1)
analogWrite(mot4,val0); // Apply the value(1) on the output of motor(2)
digitalWrite(mot3,LOW); //Apply low on the output of motor(1)
digitalWrite(mot5,LOW); //Apply low on the output of motor(2)
}
if (val1<-10) // if the value less than 10 perform if condition

```

```

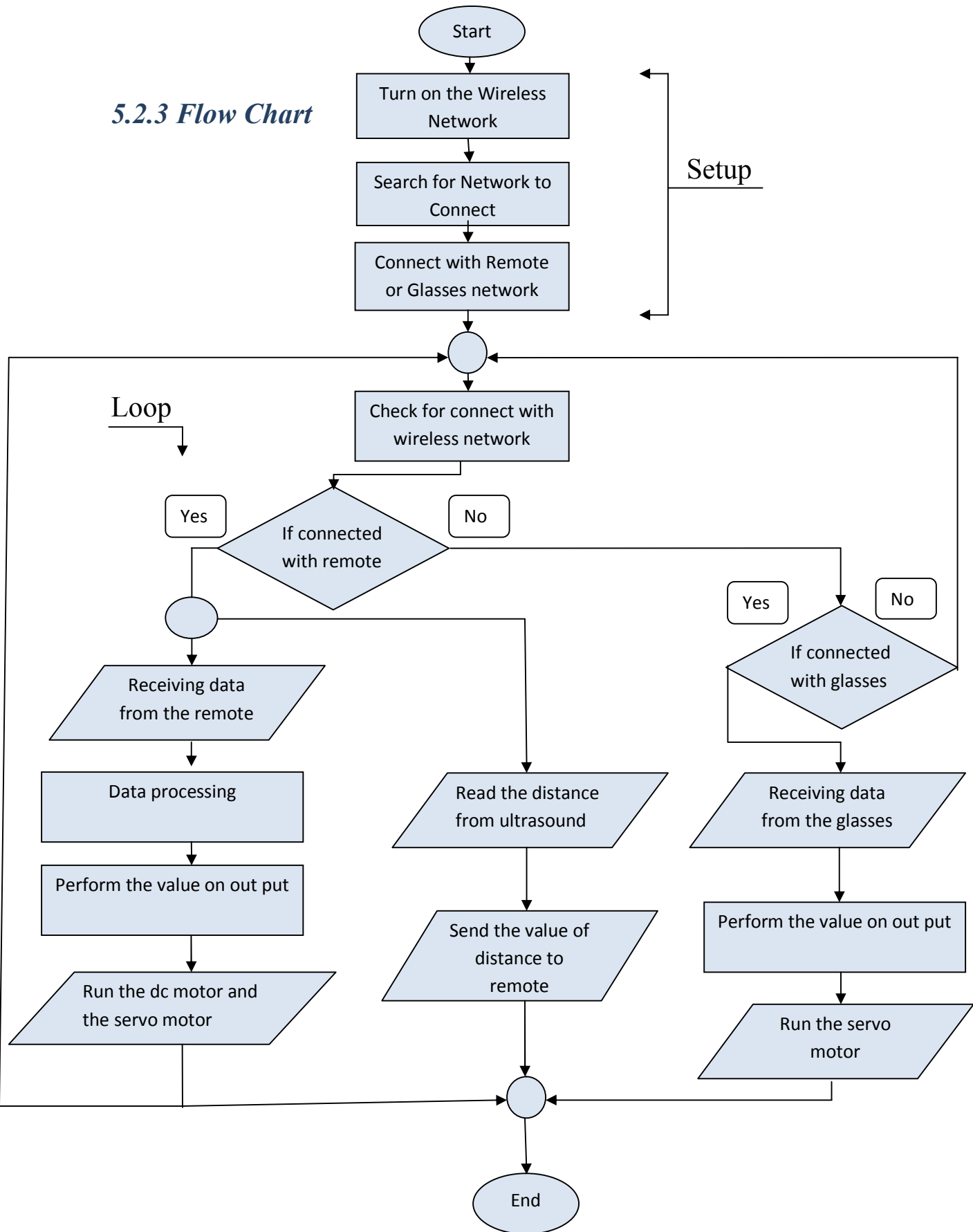
{ val1=val1*-1; // change the value to positive
analogWrite(mot3,val0); // Apply the value(1) on the output of motor(1)
analogWrite(mot5,val0); // Apply the value(1) on the output of motor(2)
digitalWrite(mot2,LOW); //Apply low on the output of motor(1)
digitalWrite(mot4,LOW); //Apply low on the output of motor(2)
}
}
else if(pipNum==2) // if the pipe line read from glasses perform if code

{int val4=map(recev[4],0,1024,50,100); //read the value from array (4) then
mapping the value from 0 - 1024 to 50 - 100
int val5=map(recev[5],0,1024,0,180); //read the value from array (5) then
mapping the value from 0 - 1024 to 0 - 180
myServo1.write(val4); // Apply the value on the output of servo1
myServo2.write(val5); // Apply the value on the output of servo2
delay(5); // delay 5 millisecond
}
delay(20); // delay 20 millisecond
wirelessSPI.stopListening(); // wireless-SPI stop listening from remote and
glasses
long duration, distance; // define variable for ultrasound parameter
digitalWrite(trigPin, LOW); //pulse low at first
delayMicroseconds(2); //delay 2 microseconds
digitalWrite(trigPin, HIGH); // start the pulse sending
delayMicroseconds(10); //delay 10 microseconds
digitalWrite(trigPin, LOW); //stop the pulse

duration = pulseIn(echoPin, HIGH); //wait to return the pulse
distance = (duration/2) / 29.1; // calculation the distance
trans[0]=distance; // storing the value of distance in transition array
wirelessSPI.write(&trans, sizeof(trans)); // sending the array to remote
control
} }

```

5.2.3 Flow Chart



5.3 Remote Control Code

5.3.1 Code Description

The Remote controls the movement of the robot and also controls the movement of the arm carrying the camera through two joysticks, The microcontroller reads the joystick values via the analog inputs and stores these values in the Array and sends it via the wireless network, The remote also reads the Array that sent by the robot contains the values of distance, reading and display on the LCD screen.

5.3.2 Remote Code

```
#include <SPI.h> //Call SPI library so you can communicate with the
nRF24L01+

#include <nRF24L01.h> // Call nRF2401 library
#include <RF24.h> // library which helps us to control the radio modem
#include <Wire.h> // library to select the wire for LCD on analog input
#include <LiquidCrystal_I2C.h> // Call liquid Crystal library for LCD
command
LiquidCrystal_I2C lcd(0x27,20,4); // define LCD pin on LCM

const int pinCE = 9; //This pin is used to set the nRF24 to standby (0) or
active mode (1)
const int pinCSN = 10; //This pin is used to set the nRF24 SPI Chip Select

int data[6]; // define array for storage the data for joystick
int ultrasound[6]; // define array for reading the data from ultrasound

RF24 wirelessSPI(pinCE, pinCSN); //set pin number of CE and CSN for
NRF24l01 chip

const uint64_t writeAddress = 0xB00B1E50D2LL; // Address for writing
pipe line in NRF
const uint64_t readAddress = 0xB00B1E50B1LL; // Address for reading
pipe line in NRF
```

```

void setup()
{
  lcd.init(); // initialize the liquid crystal LCD
  for(int i = 0; i< 3; i++) // for loop 3 time
  { lcd.backlight(); // LCD command to turn off back light
    delay (250); // delay 250 millisecond
    lcd.noBacklight(); // LCD command to turn on back light
    delay(250); // delay 250 millisecond
  }

  wirelessSPI.begin(); // turn on the wireless network
  wirelessSPI.setPALevel(RF24_PA_MIN) // set the level for amplifier power
  for modules are close to each other
  wirelessSPI.openWritingPipe(writeAddress); // open pipe line for write to
  robot and select number pipe (1)
  wirelessSPI.openReadingPipe(1,readAddress); // open pipe line for reading
  from robot and select number pipe (1)

  lcd.setCursor(0,0); // set a cursor on row number(0) and column number (0)
  lcd.print("Remote ready!"); // print message " Remote ready ! " on LCD
  delay(3000); // delay3000 millisecond
}

void loop()
{delay (20); // delay 20 millisecond
  wirelessSPI.stopListening(); // wireless-SPI stop listening from robot
  int in0 = analogRead(A0); // read the joystick for DC motor movement on
  analog pin (0) direction left and right
  int in1= analogRead(A1); // read the joystick for DC motor movement on
  analog pin (1) direction forward and backward
  int in2= analogRead(A2); // read the joystick for servo motor movement on
  analog pin (2) direction up and down
  int in3= analogRead(A3); // read the joystick for servo motor movement on
  analog pin (3) direction left and right
}

```



```

data [0]= in0; // storing the value of joystick int0 in transition data array
place number [0]
data [2]= in2; // storing the value of joystick int1 in transition data array
place number [1]
data [2]= in2; // storing the value of joystick int2 in transition data array
place number [2]
data [2]= in2; // storing the value of joystick int3 in transition data array
place number [3]

wirelessSPI.write( data, sizeof(data) ); // sending the array to robot via
wireless network
delay(20); // delay 20 millisecond
wirelessSPI.startListening(); // start wireless-SPI listening to
  wirelessSPI.read(&ultrasound, sizeof(ultrasound)); // start reading form
remote and glasses

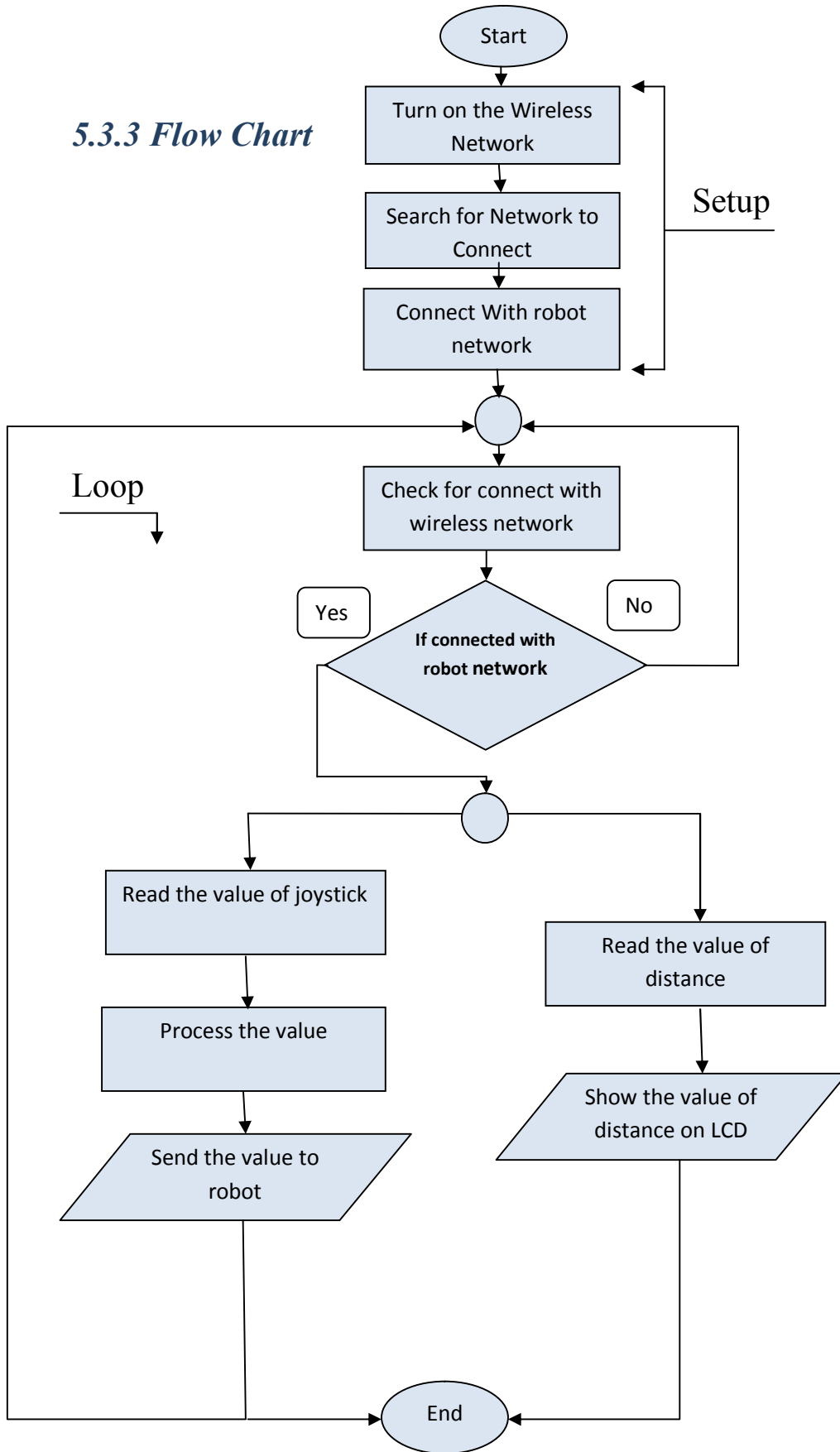
long distance; // define variable for ultrasound parameter
distance= ultrasound[0]; // reading the distance from ultrasound array

lcd.clear(); // LCD clear command
lcd.setCursor(0,0); set a cursor on row number(0) and column number (0)
lcd.print("Discovery Robot"); // print message " Discovery Robot " on LCD
lcd.setCursor(1,1); set a cursor on row number(1) and column number (1)
lcd.print("distance = "); // print message " Distance = " on LCD
lcd.setCursor(1,9); set a cursor on row number(1) and column number (9)
lcd.print(distance ); // print message the distance value on LCD
delay(50); // delay 50 millisecond

}

```

5.3.3 Flow Chart



5.4 3D-Glasses Code

5.4.1 Code Description

The 3D-Glasses display the video through a special application on the Smartphone, and control the movement of the servo motor through the MPU sensor, which changes the X and Z coordinates according to the movement of the 3D-Glasses and transfers the variables of X and Z to the robot by wireless network, where the change X expresses the horizontal movement and the variation in Z reflects the vertical movement of the 3D glasses.

5.4.2 3D-Glasses Code

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation is used in I2Cdev.h

#include "Wire.h" // library to select the wire for MPU pin on analog input
#include <SPI.h> //Call SPI library so you can communicate with MPU
#include <nRF24L01.h> // Call nRF2401 library
#include <RF24.h> // library which helps us to control th eradio modem

const int pinCE = 9; //This pin is used to set the nRF24 to standby (0) or
active mode (1)
const int pinCSN = 10; //This pin is used to set the nRF24 SPI Chip Select

int data[6]; define array for storage the data for MPU

RF24 wirelessSPI(pinCE, pinCSN); //set pin number of CE and CSN for
NRF24l01 chip
```

```

const uint64_t writeAddress = 0xB00B1E50C3LL; // Address for writing
pipe line in NRF
const uint64_t readAddress = 0xB00B1E50A4LL; Address for reading pipe
line in NRF
#endif

MPU6050 mpu; //use for AD0 high- AD0 high = 0x69
#define OUTPUT_READABLE_YAWPITCHROLL //uncomment
"OUTPUT_READABLE_REALACCEL" if you want to see acceleration
components with gravity

// MPU control status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 =
success, !0 = error)

uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorInt16 aa; // [x, y, z] accel sensor measurements
VectorInt16 aaReal; // [x, y, z] gravity-free accel sensor measurements
VectorInt16 aaWorld; // [x, y, z] world-frame accel sensor measurements
VectorFloat gravity; // [x, y, z] gravity vector
float euler[3]; // [psi, theta, phi] Euler angle container
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector
// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0,0, 0,0, 0x00, 0x00, '\r', '\n' };
volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin
has gone high
void dmpDataReady() {
    mpuInterrupt = true; }

```

```

void setup() {
  wirelessSPI.begin(); //turn on the wireless network
  wirelessSPI.openWritingPipe(wAddress); // open pipe line for write to robot
  and select number pipe (2)
  wirelessSPI.stopListening();// wireless-SPI stop listening from robot
  // join I2C bus (I2Cdev library doesn't do this automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
  #endif
  mpu.initialize();
}
void loop() {
  // if programming failed, don't try to do anything
  if (!dmpReady) return;
  // wait for MPU interrupt or extra packet(s) available
  while (!mpuInterrupt && fifoCount < packetSize) { } // if you are really
  paranoid you can frequently test in between other
  stuff to see if mpuInterrupt is true, and if so, "break;" from the while() loop
  to immediately process the MPU data
  mpuInterrupt = false; // reset interrupt flag and get INT_STATUS byte
  mpuIntStatus = mpu.getIntStatus();
  // get current FIFO count
  fifoCount = mpu.getFIFOCount();
  // check for overflow (this should never happen unless our code is too
  inefficient)
  if ((mpuIntStatus & 0x10) || fifoCount == 1024)
  { // reset so we can continue cleanly
    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));}
  } else if (mpuIntStatus & 0x02)
  while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
  // read a packet from FIFO

```

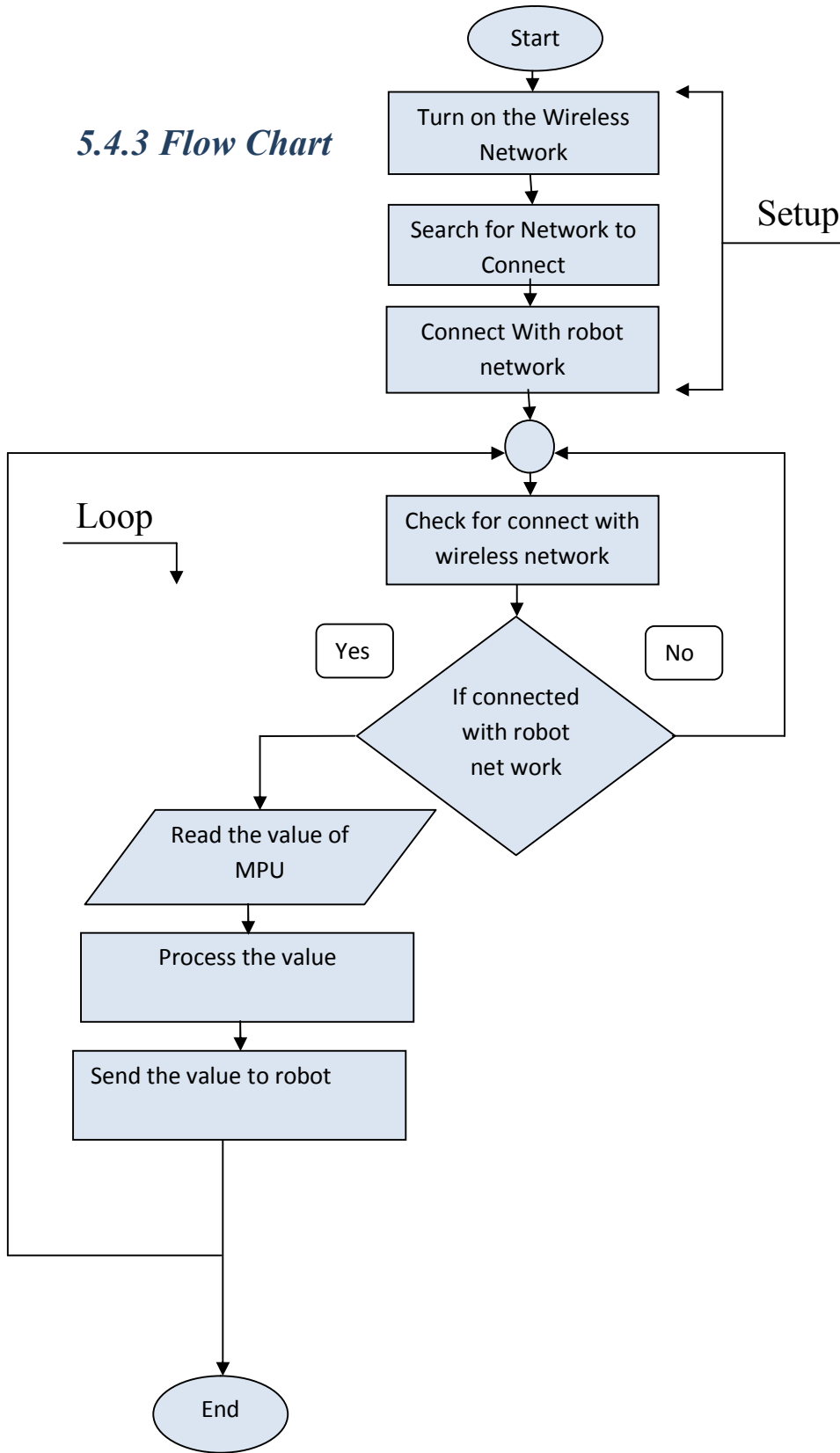
```

mpu.getFIFOBytes(fifoBuffer, packetSize);
// track FIFO count here in case there is > 1 packet available
// (this lets us immediately read more without waiting for an interrupt)
fifoCount -= packetSize;
#ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
// display real acceleration, adjusted to remove gravity
    int z= ypr[0] * 180/M_PI + (76);
    int in4 = map(z,0,180,0,1024);
    data[4]= in4; ; // storing the value of joystick int4 in transition data
array place number [4]
    int x= ypr[1] * 180/M_PI + (100);
    int in5 = map(x,0,180,0,1024);
    data[5]= in5; ; // storing the value of MPU int5 in transition data
array place number [5]
    wirelessSPI.write( data, sizeof(data) );
delay(25); #endif
#ifdef OUTPUT_TEAPOT
    // display quaternion values in InvenSense Teapot demo format:
    teapotPacket[2] = fifoBuffer[0];
    teapotPacket[3] = fifoBuffer[1];
    teapotPacket[4] = fifoBuffer[4];
    teapotPacket[5] = fifoBuffer[5];
    teapotPacket[6] = fifoBuffer[8];
    teapotPacket[7] = fifoBuffer[9];
    teapotPacket[8] = fifoBuffer[12];
    teapotPacket[9] = fifoBuffer[13];
    Serial.write(teapotPacket, 14);
    teapotPacket[11]++; // packetCount, loops at 0xFF on purpose

#endif } }

```

5.4.3 Flow Chart



Conclusion

We can conclude from this project that there are many applications on the robot, through the microcontroller can build a lot of applications through which to work on the development of the project and increase applications that the robot will be able to work on them in an easy and inexpensive.

There are also many ways to control the remote robot, including the remote control, and you can design a special application on the smartphone we can control the robot, and that has been identified methods of telecommuting in full details and ways to send and receive the data and processing of the technology used in The remote control, the distance through which it is controlled, and a lot of settings.

And calculations of all the devices mounted on the robot, including the collection of loads and determine the appropriate battery and calculate how long the battery will be able to operate, and calculate the distances by the speed of sound in the ultrasound technology, and other calculations of speed and torque of small electric motors.

In the end, it can be said that the Discovery Robot is a complete set of sensors and control devices and power devices that operate in a integrated and programmed system, and we were able to use what we learned and concluded in the courses we learned the previous years.

References

- i. [1] *Ansorge-John, WEBBOT, University of Nebraska, (July, 2006)*
- ii. [2] *NASA - MER-B Mars Exploration Rover.*
- iii. [3] *Arduino -<https://www.arduino.cc>.*
- iv. [4] *Michael Wrigh, Using the NRF24101 2.4ghz Rf Control Module*
- v. [5] *Al Williams (2002), Microcontroller projects using the Basic Stamp, (2nd ed.).*
- vi. [6] *Herman, Stephen L. Electric Motor Control. 9th ed. Delmar, Cengage Learning, 2009.*
- vii. [7] *Stephen J. Dodds, Servo Motors and Industrial Control Theory,2014*
- viii. [8] *Jon Lazar, Arduino and LEGO Projects, 2013*
- ix. [9] *Masaki Yoshio, Ralph J. Brodd, Akiya Kozawa, Lithium-Ion Batteries,2010*
- x. [10] *Lentin Joseph, Learning Robotics Using Python, May 2015*

