Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN1093 – Laboratory Practice I

Hand Gesture Controlled Robot

Group number: 28

Jathurshan P.	- 170248G
U.G.C. Jayasankha	- 170259P
S.N. Liyanagoonawardena	- 170348M
Mithunjha A.	- 170389M

This is submitted as a partial fulfillment for the module EN1093: Laboratory Practice I

Department of Electronic and Telecommunication Engineering, University of Moratuwa

10th January 2019

Abstract

The aim of this device is to easily navigate moving objects with minimum effort and be used as a simulation for large moving objects found in real life. This device is extremely useful for the users who have certain disabilities, lack of physical strength and suffer from fatigue. Currently the industry lacks an automated, low cost solution that is very simple to operate and depend solely on manual labor. In this project we control the machine with the movement of hand which simultaneously controls the movement of robot. The device consists mainly of two parts, the transmitter and the receiver. The transmitter consists of an accelerometer to detect the direction via a tilt of the hand and the receiver is included with an LCD display to indicate the direction and a DC motor driver. Both transmitter and the receiver uses Atmel microcontrollers. The communication between the two parts is done using Bluetooth modules. As we successfully obtained the expected results and considering its efficiency, low cost and simplicity this project can be expected of a vast market demand with a bit more improvement and a professional touch.

Contents

1. Introduction

1.1 Overview

The hand gesture controlled robot is used to move any device with wheels, using minimum physical labor, in the direction given simply by a tilt of the hand. This device could be used in hospitals where equipment such as the patient wheelchairs, hospital beds and medical device holders are be moved frequently. Thus, this product reduces the effort, man power and hence cost. This device could also be utilized in different such as workplaces, garments and supermarkets where large quantities of heavy loads are constantly moved. This could also serve in daily household tasks such as moving prams and certain furniture.

1.2 Algorithm

The following figure shows the sequence of steps used in the transmitter.

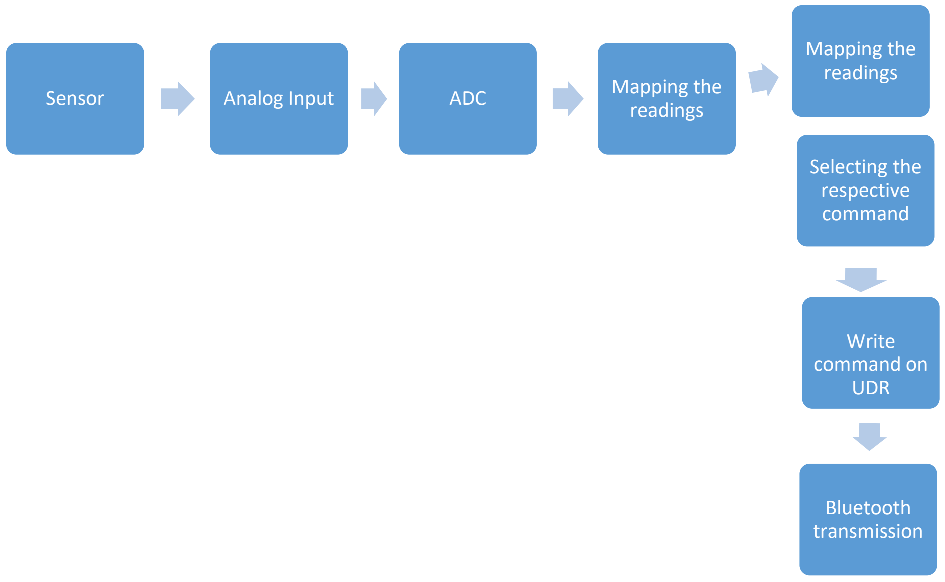

Fig. 1.1 The algorithm of the transmitter

The following figure shows the sequence of steps used in receiver.

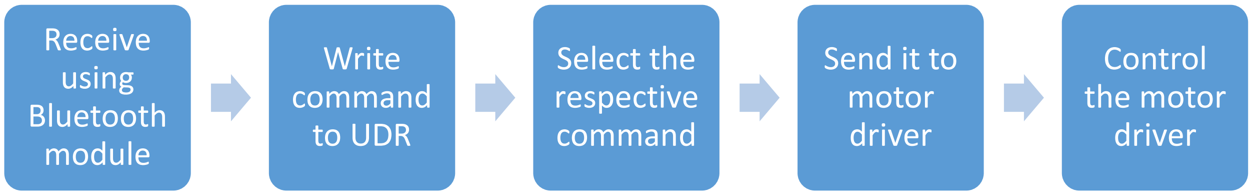

Fig. 1.2 The algorithm of the receiver

1.3 The final outlook

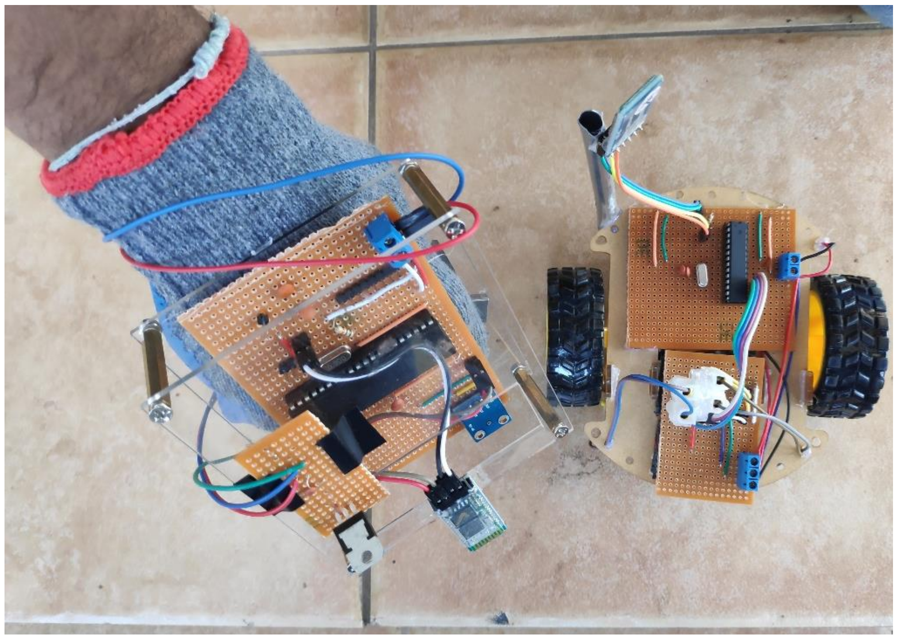

Fig. 1.3.1 the transmitter and the receiver

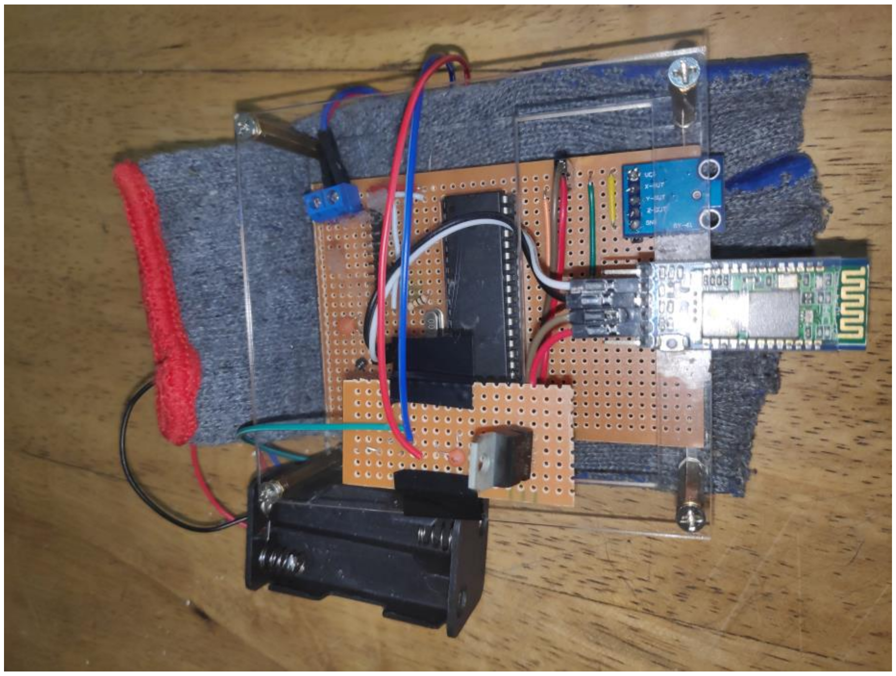
Fig. 1.3.2 the transmitter

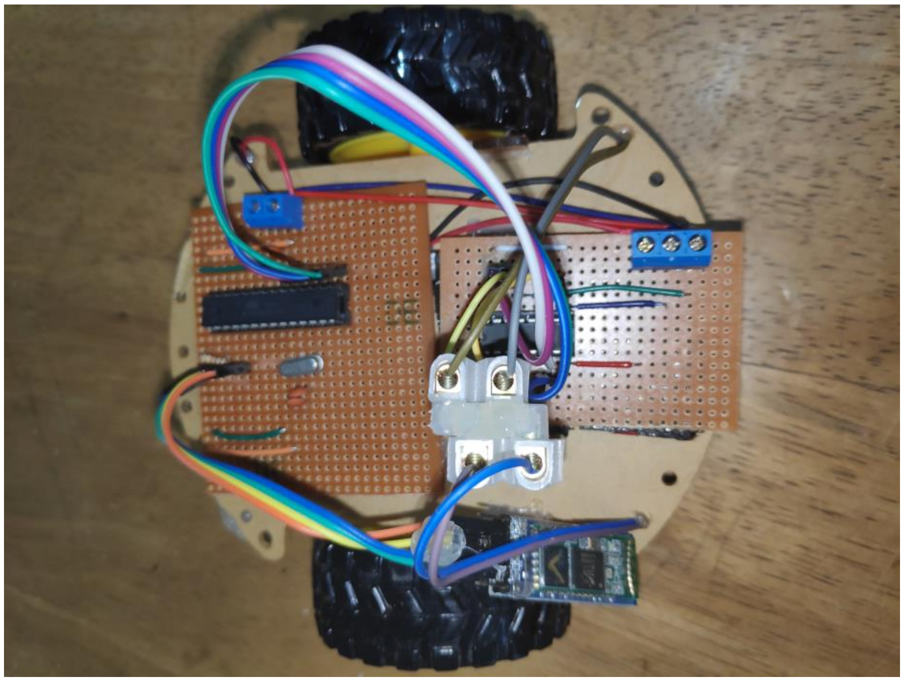
Fig. 1.3.3 the receiver

2. Method

2.1 Description

This device will use hand gestures as input signals to drive the robot in different direction and display the direction of movement of the robot in an alphanumeric LCD. Hand gestures are given by the tilt of hand in different direction. An accelerometer sensor is used to detect different hand gestures. The sensor will be attached to our hand through a hand glove. After converting the analog signals of accelerometer sensor to digital values, the microcontroller will be used to process the digital values to find different gestures of the hand. Once the hand gesture is known, the Atmel microcontroller will send the required signal to the DC motor driver of the robot to drive the robot in the desired direction

2.2 Circuit in the transmitter

2.2.1 Operation

The transmitter measures the tilt of the accelerometer attached to the glove and the LCD display displays the direction according to the measured reading and its respective string given in the code. There are four directions with respect to four different hand gestures defined by the code. They are,

1. Forward tilt – To move the device in the forward direction
2. Backward tilt – To move the device in the backward direction
3. Right tilt – To move the device in the right direction
4. Left tilt – To move the device in the left direction

The micro controller decides which the respective condition to the given direction is and it sends the corresponding signal to the Bluetooth transmitter.

2.2.2 Components used

- ATMEGA 32A micro controller
- LCD display
- Accelerometer
- Potentiometer
- Oscillator
- 22pF capacitors -3
- 1 kilo ohm resistor
- 10 kilo resistor

2.2.3 Basic configuration of the micro controller [1]

- Pin 9 –Reset pin, is connected to the 5V power supply through a 1k resistor
- Pin 10, Pin 30 – VCC
- Pin 11, Pin 31, Pin 32 – Ground
- Pin 12, Pin 13– Connected to the 16MHz oscillator and two 22pF capacitors

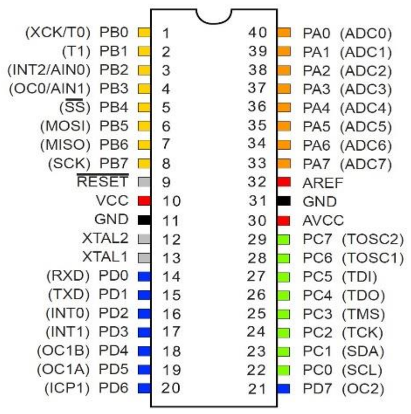

Fig. 2.1 Pin configuration of ATMEGA 32A [5]

2.2.4 LCD display configuration

- Pin 1 – VSS, Ground
- Pin 2 – VDD, 5V
- Pin 3 – VEE, Connected to 5V through a 2k resistor
- Pin 4 – RS, Connected to pin 16 (D2) of the micro controller
- Pin 5 – RW, Connected to pin 17 (D3) of the micro controller
- Pin 6 – E, Connected to pin 18 (D4) of the micro controller
- Pin 7,8,9,10,11,12,13,14 – Connected to the pins 1-8 in the micro controller

2.2.5 Accelerometer and LCD configuration [2]

- Pin X – Pin 40 of the micro controller
- Pin Y – Pin 39 of the micro controller
- Ground – Connected to the ground of the circuit

- VCC – 5V

2.2.6 Bluetooth module configuration

- TX port – Pin 13
- RX port – Pin 14
- VCC pin – 5V
- GND pin – Connected to ground

2.2.7 Circuit Diagram

2.2.7.1 Schematic

The following diagram shows the schematic of the transmitter.

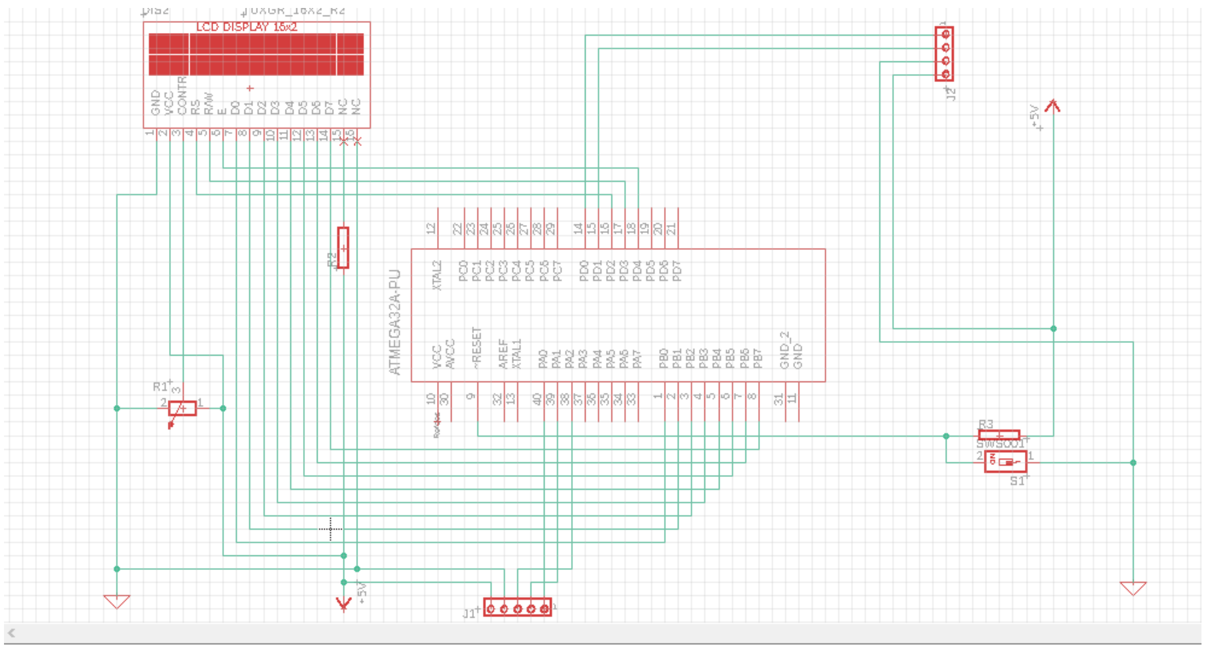

Fig 2.2 the schematic of the transmitter

2.2.7.2 Layout

The following diagram shows the layout of the transmitter

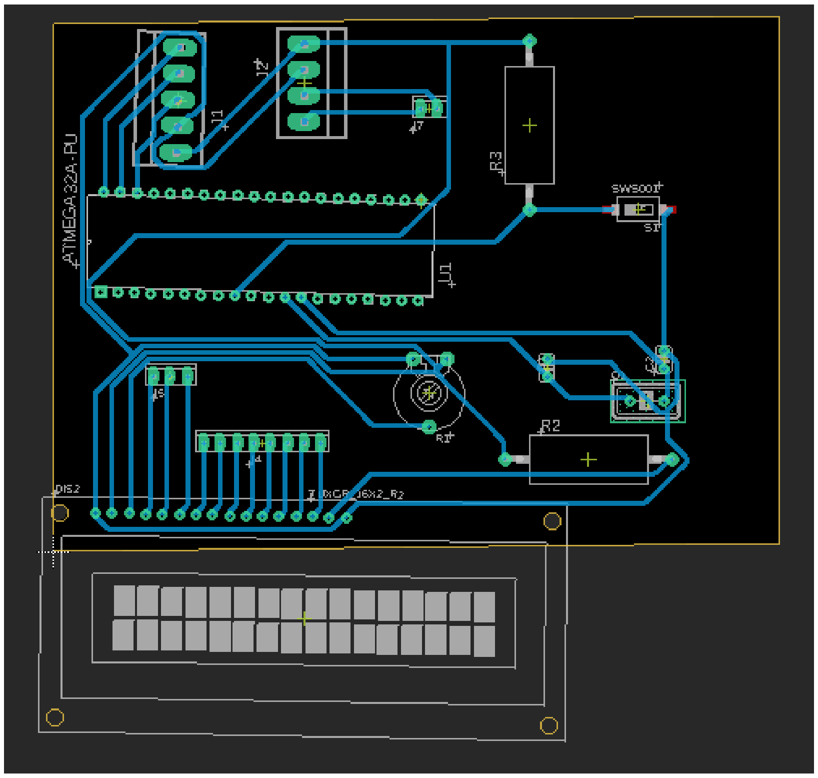

Fig 2.3 the layout of the transmitter

2.2.8 Code

The code is included in the appendices. [I]

2.3 Circuit in the receiver

2.3.1 Operation

The receiver accepts the signal sent from the transmitter and moves the corresponding wheels and their speeds according to the code resulting in moving the car in-line with the corresponding gesture given by the hand.

2.3.2 Components used

- ATMEGA 8A-PU micro controller
- Oscillator
- 22pF capacitors -3
- DC motor
- 1 kilo ohm resistor
- Wheels
- L293D driver

2.3.3 Basic configuration of the micro controller

- Pin 1 – Reset
- Pin 2, 3 – Connected to Bluetooth
- Pin 15,16,17,18– Connected to L293D
- Pin 21– AREF, Connected to the ground through a 10uF capacitor
- Pin 8,22 – Connected to ground
- Pin 20 - AVCC, Connected to 5V
- Pin 7- VCC, Connected to 5V

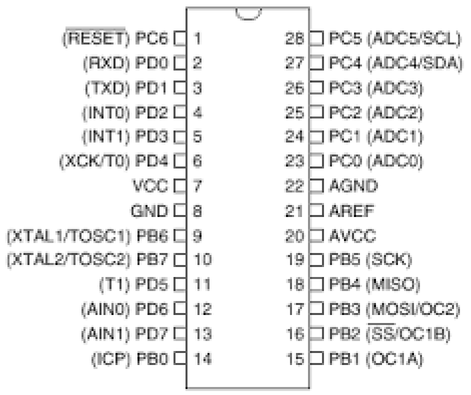

Fig 2.4 Pin configuration of ATMEGA 8PU [6]

2.3.4 L293D configuration

- Pin3, Pin 6 – Outputs, Right wheel
- Pin 11, Pin 14- Outputs, Left wheel
- Pin 16 - VSS, 5V

- Pin 8 – VS, 12V
- Pin 1,9– 5V
- Pin2 – Connected to the pin 22 of the micro controller
- Pin 7 – Connected to the pin 23 of the micro controller
- Pin 10 – Connected to the pin 24 of the micro controller
- Pin 15 – Connected to the pin 25 of the micro controller
- Pin 4,5,12,13- Connected to ground

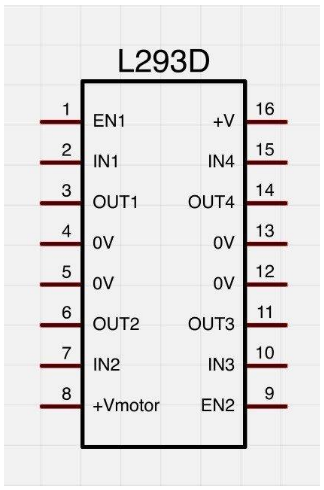

Fig 2.5 Pin configuration of L293D [7]

2.3.5 Bluetooth module configuration

- TX pin – Pin 2
- RX pin – Pin 3
- VCC pin – 5V
- GND pin – Connected to ground

2.3.6 Circuit Diagram

2.3.6.1 Schematic

The following diagram shows the schematic of the transmitter.

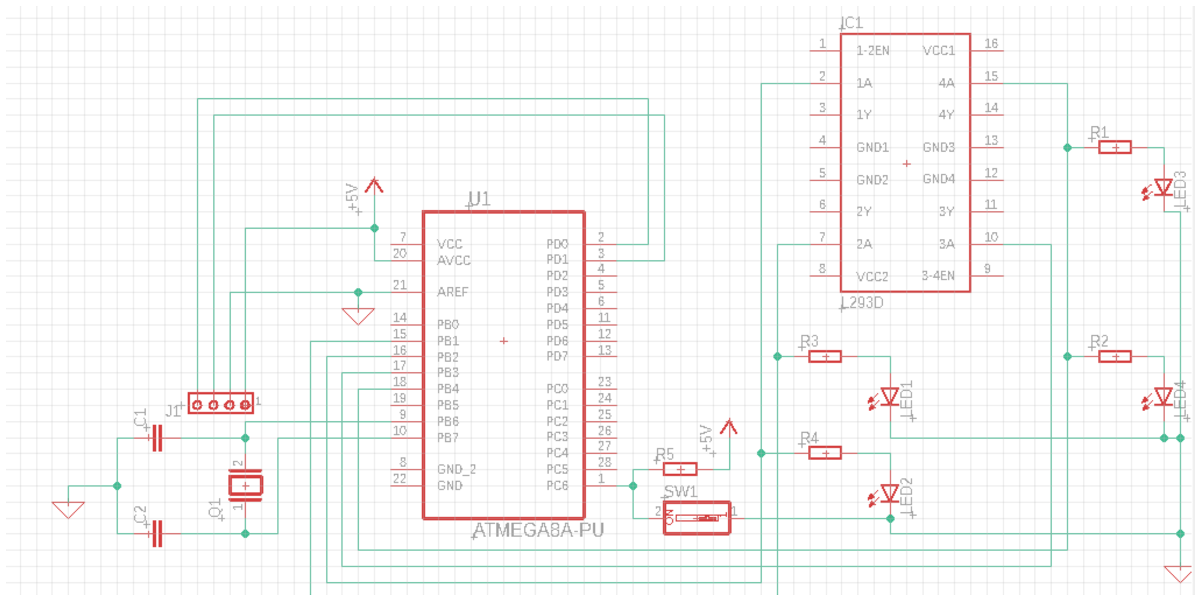

Fig 2.6 the schematic of the receiver

2.3.6.2 Layout

The following diagram shows the layout of the receiver.

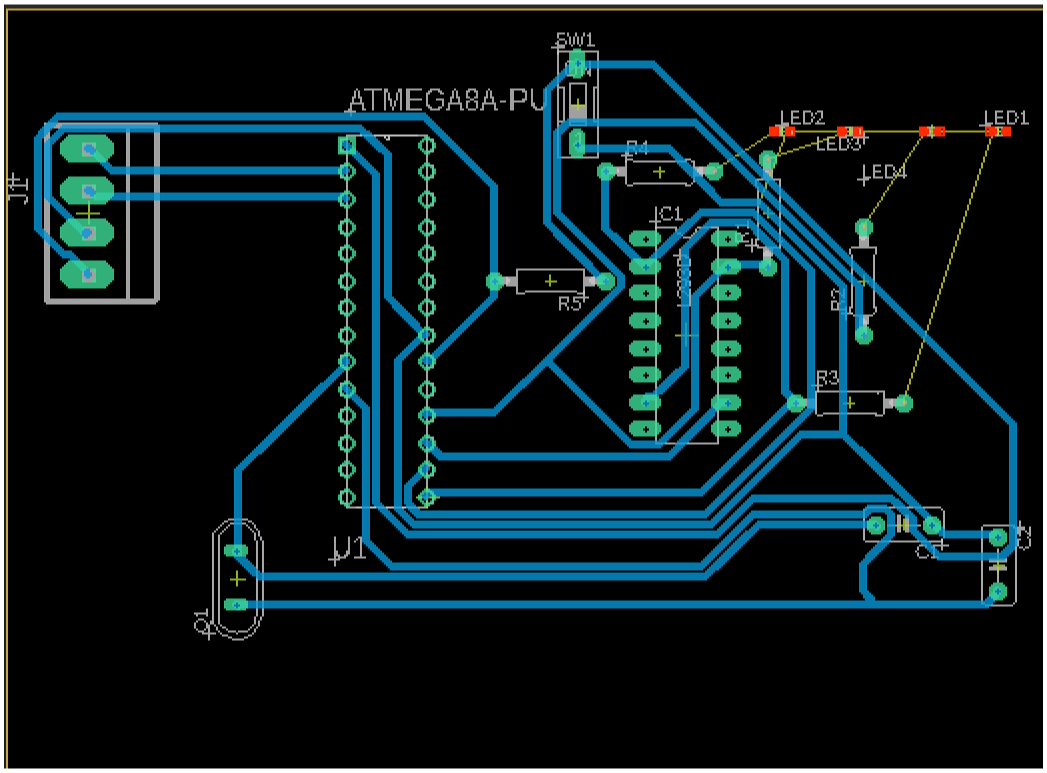

Fig 2.7 the layout of the receiver

2.3.7 Code

The code is included in the appendices. [III]

2.4 Testing Procedure

2.4.1 Process and the issues encountered

1. Making the circuit to upload the code and uploaded the code using USBasp driver.
2. Testing the circuit and the code by blinking a LED.
3. Testing the coding for the LCD display by displaying a short string, but failed to display the desired string.
4. Successfully displayed the desired string.

5. Testing the accelerometer and LCD display together to display the change of readings, with the tilt, in the accelerometer. But the readings didn’t change with the tilt.
6. Successfully displayed the change in readings with the tilt.
7. Successfully mapped the readings from -90 to +90 which were previously 0 to 255
8. When tried to turn the wheels using the motor driver, 1 wheel didn’t turn.
9. After successfully turned both wheels, connecting the battery didn’t make any changes to the speed of rotation
10. Successfully turned both wheels and managed their speeds.
11. Taking the readings from the accelerometer in 4 directions and coding the wheels to be rotated according to the readings. But failed as 1 wheel continuously rotated in the same speed and didn’t turn according to the readings.
12. After successfully overcoming the previous issue 1 wheel kept stopping in intervals while other kept rotating continuously.
13. Successfully managed to rotate both wheels continuously with the accelerometer readings.
14. Testing the Bluetooth module by blinking a bulb using the phone.
15. Successfully paired the Bluetooth devices by master slave.
16. Tested the HC05 to HC05 transmission but what received was garbage.
17. Transmitted 1 and 0 to be displayed in the LCD display but failed as it displayed 2 random symbols. (garbage data)
18. Successfully managed to transmit the desired commands from the transmitter to receiver.

2.4.2 Solutions to the issues

1. In the beginning coding the project using Atmel was a challenge as it was the 1st time using the software. But this issue was overcome by learning the basics of the software.
2. The failure to display the desired string in the LCD display was a soldering issue of the LCD where 1 pin was not properly connected. Hence we soldered the LCD display again and solved the issue.
3. The reason why the accelerometer reading didn’t change with the tilt was because the circuit lacked capacitor in AREF pin of the microcontroller. Hence by adding a 22pF capacitor this issue was overcome.
4. Initially the failure of one wheel to be turned was because 1 pin was broken in the L293D, hence we replaced the IC

5. The reason for the failure of the battery to support the wheels initially was later found out that the wire connected to the battery has been damaged from inside. So we replaced the wire fixing the problem.
6. The reason why one wheel didn't change with the accelerometer readings is because one of the pins connected to that wheel is a pre-defined hard coded pin where the user defined code cannot change its voltage. Hence we change the pins connected to the wires of that wheel.
7. Lacks of pins utilized in the circuit was the reason why 1 wheel kept stopping in intervals. Hence we used the pins of LCD display.
8. To solve the issue with garbage data we increased the clock speed of the ATMEGA 32A.

3. Results

The following table shows the percentage of ideality of the movement of the robot in the desired direction on three trials.

	TRIAL 1	*TRIAL 2*	*TRIAL 3*	*Final*
FORWARD	100%	70%	100%	100%
BACKWARD	0%	70%	0%	100%
LEFT	0%	50%	100%	100%
RIGHT	0%	50%	100%	100%

Table 3.1 the percentage of ideality of the movements

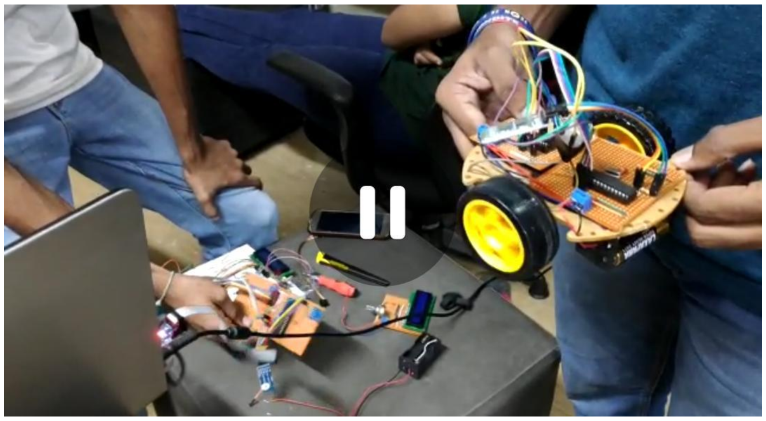
Fig 3.1 the final outcome

4. Discussions

Considering the first trial, after testing we conducted the connectivity test and realized there were some issues due to poor soldering. After adjusting and correcting them we also thought that there could be issues due to the Vcc of the accelerometer sensor. Because the sensor reading varies according to Vcc value. Atmega 32A was coded according to values produced by the accelerometer at 5V Vcc. So Voltage should be regulated to 5V. We tested again by powering up the transmitter using USBasp cable from Laptops.

Considering the second trial, the robot moved all four directions but the transition was not smooth. We thought that it could be because of the high sensitivity of accelerometer sensor. So by coding we increased the threshold value of the mapped reading. Also we made a regulator circuit to regulate the Vcc to 5V.then tested it again.

In the third trial robot moved all four directions except backwards. This occurred because the voltage regulated was around 4.2V when we gave a 6V power supply. So we thought about trying 9V battery for the transmitter or take the readings and map the values again. We tested using a 9V battery and the robot moved all four directions smoothly. We can further improve this product by compacting the components more and decreasing its size. Also this can be extended more in the future by adjusting it so that the user can have more control over the speed of the robot.

5. Acknowledgement

A special thanks goes out to our instructor Mr. Dulara De Zoysa for the guidance and support rendered and the instructors Mr. Asanga and Mr. Janith Gunarathne who constantly assisted us in overcoming issues and encouraged us for self-studies. Also we express our gratitude to Prof. (Mrs.) Dileeka Dias and Dr. Ranga Rodrigo for their constant guidance in writing a successful report. Also we are grateful for Dr. Jayathu Samarawickrama for his guidance and encouragement offered. We would also like to express our gratitude to the staff of ENTC department without whom we could not accomplish the project.

6. References

[1] Microchip, "Microchip ATmega32A," [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega32A .

[2] "Robotics Radio control Power electronics," Dimension Electronics, [Online]. Available: https://www.dimensionengineering.com/info/accelerometers.

[3] R. Barnett and L. C. S. O'cull, Embbed C programming and the Atmel AVR, New York: Thompson, 2007.

[4] A. Chaudry, Robust hand gesture recognition for robotic hand control, Gateway East: Springer, 2018.

[5] J. S. C. Bray, Bluetooth, Delhi: Pearson, 2002.

[6] "AtmegaA8," [Online]. Available: https://avrhelp.mcselec.com/index.html?m8.htm.

[7] "L293D Datasheet," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/L293D-Datasheet-9_fig2_301443618.

7. Appendices

[I] the code for the transmitter

```
/*

 * Transmitter Atmega32A.c

 *

 * Created: 1/5/2019 5:02:48 PM

 * Author: Group 28

 */
```

```
#define F_CPU 8000000                // set the Clock Speed of the ATmega 32A

#define BAUD 9600                   // set the baud rate for HC05 to HC05 data transmission

#define MYUBRR F_CPU/16/BAUD-1       //configure the baud rate in the ATmega 32A....equation for
Baud rate setting


#define D0 eS_PORTB0                //Defining the pins for the LCD Display

#define D1 eS_PORTB1

#define D2 eS_PORTB2

#define D3 eS_PORTB3

#define D4 eS_PORTB4

#define D5 eS_PORTB5

#define D6 eS_PORTB6

#define D7 eS_PORTB7

#define RS eS_PORTD2

#define EN eS_PORTD4




#include <avr/io.h>

#include <util/delay.h>           //library to implement delays

#include <lcd.h>                 //library for LCD Display


void USART_Init( unsigned int ubrr)         //Initiate USART
```

```
{
     UBRRH = (unsigned char)(ubrr>>8);       // Set baud rate
    UBRRL = (unsigned char)ubrr;            // Enable receiver and transmitter

    UCSRB = (1<<RXEN)|(1<<TXEN);            // Set frame format: 8data, 2stop bit
    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}

void USART_Transmit( unsigned char data )           //Function to transmit unsigned char through
bluetooth
{
    while ( !( UCSRA & (1<<UDRE)) )          // Wait for empty transmit buffer
    ;                                  // Put data into buffer, sends the data
    UDR=data;
}

unsigned char USART_Receive( void )           //Function to receive unsigned char through bluetooth
{
    while ( !(UCSRA & (1<<RXC)) )            // Wait for data to be received
    ;

    // Get and return received data from buffer
    return UDR;
}
```

```
void Adc32init()                    // function to initiate Analog to Digital conversion ports

{

    DDRA= 0b00000000;               //set PORT A as input port

    ADMUX=0b01100000;

    ADCSRA = (1<<ADEN);


}


float ADXLread(int a)               //Function to read analog input

{

    int x;

    ADMUX=0b01100000 + a;           //set the specific input port in ADC ports

    ADCSRA = ADCSRA|(1<<ADSC);      //start ADC

    while (ADCSRA & (1<<ADSC));

    x=ADCH;                         //assign x as converted value in ADCH registors

    return x ;


}



void itoa(int __val, char *__s, int __radix);     //integer to char conversion

float x_axis,y_axis;
```

```
int main(void)

{

    DDRB=0xff;                //set PORTB as output ports

    DDRD=0xff;                //set PORTD as output ports



    Adc32init();              //intiate ADC

    USART_Init(MYUBRR);       //initiate USART



    Lcd4_Init();              //Initiate LCD

    Lcd4_Set_Cursor(1,1);

    Lcd4_Write_String("Hand Gesture");

    Lcd4_Set_Cursor(2,1);

    Lcd4_Write_String("Controlled Robot");

    _delay_ms(1000);

    Lcd4_Clear();

    Lcd4_Set_Cursor(1,1);

    Lcd4_Write_String("By Group ");

    Lcd4_Set_Cursor(2,1);

    Lcd4_Write_String("55 & 56");

    _delay_ms(1000);

    Lcd4_Clear();
```

```
/* Replace with your application code */

while (1)

{

        x_axis=ADXLread(0);             //take reading of x axis from ADXL335

        x_axis= ((x_axis-62)/41)*180-90;     //map the value to -90 to +90

        char *x="00000";

        itoa(x_axis,x,10);

        Lcd4_Set_Cursor(1,1);            //display the value on LCD

        Lcd4_Write_String("x:");

        Lcd4_Set_Cursor(2,1);

        Lcd4_Write_String (x);



        y_axis=ADXLread(1);           //take reading of y axis from ADXL335

        y_axis = ((y_axis-62)/34)*180-90;  //map the value to -90 to +90

        char *y="00000";

        itoa(y_axis,y,10);

        Lcd4_Set_Cursor(1,8);          //display the value on LCD

        Lcd4_Write_String("y:");

        Lcd4_Set_Cursor(2,8);

        Lcd4_Write_String(y);
```

```
if (y_axis<-50){

        USART_Transmit('1');

        //Robot will move in forward direction



        }else if(y_axis>50){

        USART_Transmit('2');

        //Robot will move in reverse direction

        }else if(x_axis>50){

        USART_Transmit('3');

        //Robot will move in right direction



        }else if(x_axis<-50){

        USART_Transmit('4');

        //Robot will move in left direction



        }else{

USART_Transmit('0');

        //Robot will stop , ALL ARE LOW



}



//_delay_ms(1000);
```

```
            Lcd4_Clear();    //clear the LCD


   }

}
```

[II] The code for the receiver

```
/*

 * Receiver Atmega8A.c

 *

 * Created: 1/5/2019 4:38:11 PM

 * Author : Group 28

 */

#define F_CPU 8000000        // set the internal Clock Speed of the Atmega8a

#define BAUD 9600            //set the baud rate of the transmission

#define MYUBRR F_CPU/16/BAUD-1    //calculate to set baud rate in the chip

#include <avr/io.h>

#include <util/delay.h>            //to use delays




void USART_Init( unsigned int ubrr)     //Initiate USART

{

    /* Set baud rate */

    UBRRH = (unsigned char)(ubrr>>8);
    UBRRL = (unsigned char)ubrr;

    /* Enable receiver and transmitter */

    UCSRB = (1<<RXEN)|(1<<TXEN);

    /* Set frame format: 8data, 2stop bit */

    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);

}


void USART_Transmit( unsigned char data )    //Function to transmit unsigned char

{

    /* Wait for empty transmit buffer */

    while ( !( UCSRA & (1<<UDRE)) )

    ;

    /* Put data into buffer, sends the data */

    UDR=data;

}


unsigned char USART_Receive( void )        //Function to receive data

{

    /* Wait for data to be received */

    while ( !(UCSRA & (1<<RXC)) )

    ;

    /* Get and return received data from buffer */

    return UDR;
```

```
}


unsigned char a;

int main(void)

{

      DDRB=0xff;                  //Activate PORTB as output

      USART_Init(MYUBRR);          //initiate  usart

   /* Replace with your application code */

   while (1)

   {

                  a=USART_Receive();     //assign a to received data

                  if (a=='1'){

                              PORTB=0b00001010;     //move forward

                              //_delay_ms(1000);

                  }

                  else if(a=='2'){

                  PORTB=0b00010100;     //move backward

                  //_delay_ms(1000);

                  }

                  else if(a=='3'){

                              PORTB=0b00001000;   //move right

                              //_delay_ms(1000);

                  }
```

```
                    else if(a=='4'){

                            PORTB=0b00000010;    //move left

                            //_delay_ms(1000);

                    }else{

                            PORTB=0b00011110;  //stop

                    }


          }

        }
```