Ahmedabad University

School of Engineering and Applied Science

Embedded System Design

Final Project Report

Gesture Controlled RoboCar

Group No.:19

Group Members:

Neel Patel (201501075)

Prayer Yadav (201501124)

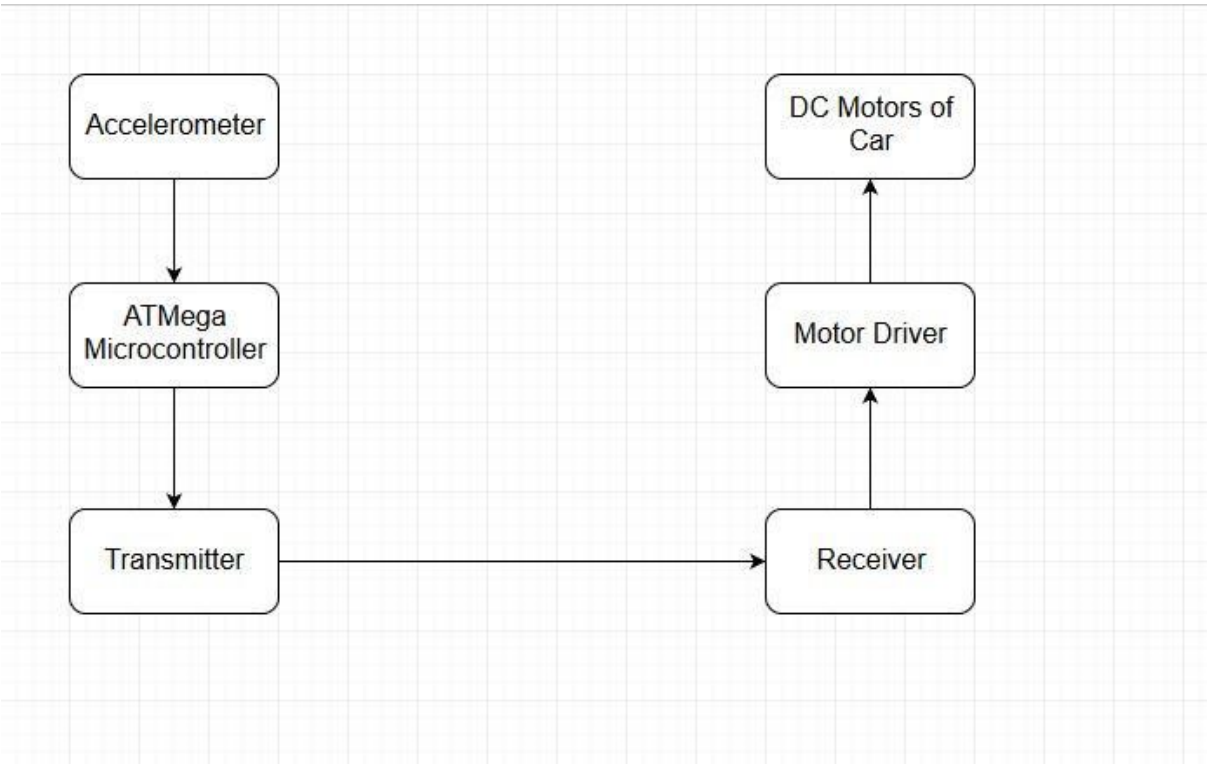
Summary:

Motivation: Technology is present in our lives on daily basis. Microcontrollers play important part in our life. Most electronic devices have some microcontroller in it. We have learned many concepts of microcontrollers especially ATMega. Thus, to implement those concepts we had to we are making gesture controlled RoboCar. We are not only using ATMega but also some other devices including accelerometer sensor etc. Thus, in this project we will also be able to learn about the connections of ATMega controller with external world. We will be able to know the functioning of sensors involved in the project.

Description: Gesture Controlled RoboCar is a robot which can be controlled by human gestures. The user just needs to wear a gesture device (hand glove) in which a sensor is included. The sensor will record the movement of hand in a specific direction which will result in the motion of the car in the respective directions. We can control the car using accelerometer sensors connected to a hand glove. Accelerometer sensor will allow the user to control the throttle of the car. Movement of car is controlled by mechanism which involves rotation of forth and rear wheels using motors.

Final Outcome: Gesture Controlled RoboCar can be controlled by user based on his gestures.

Block Diagram:



Basic Components Needed:

1) ATMega Microcontroller

2) Accelerometer Module ADXL335

3) HT12D and HT12E

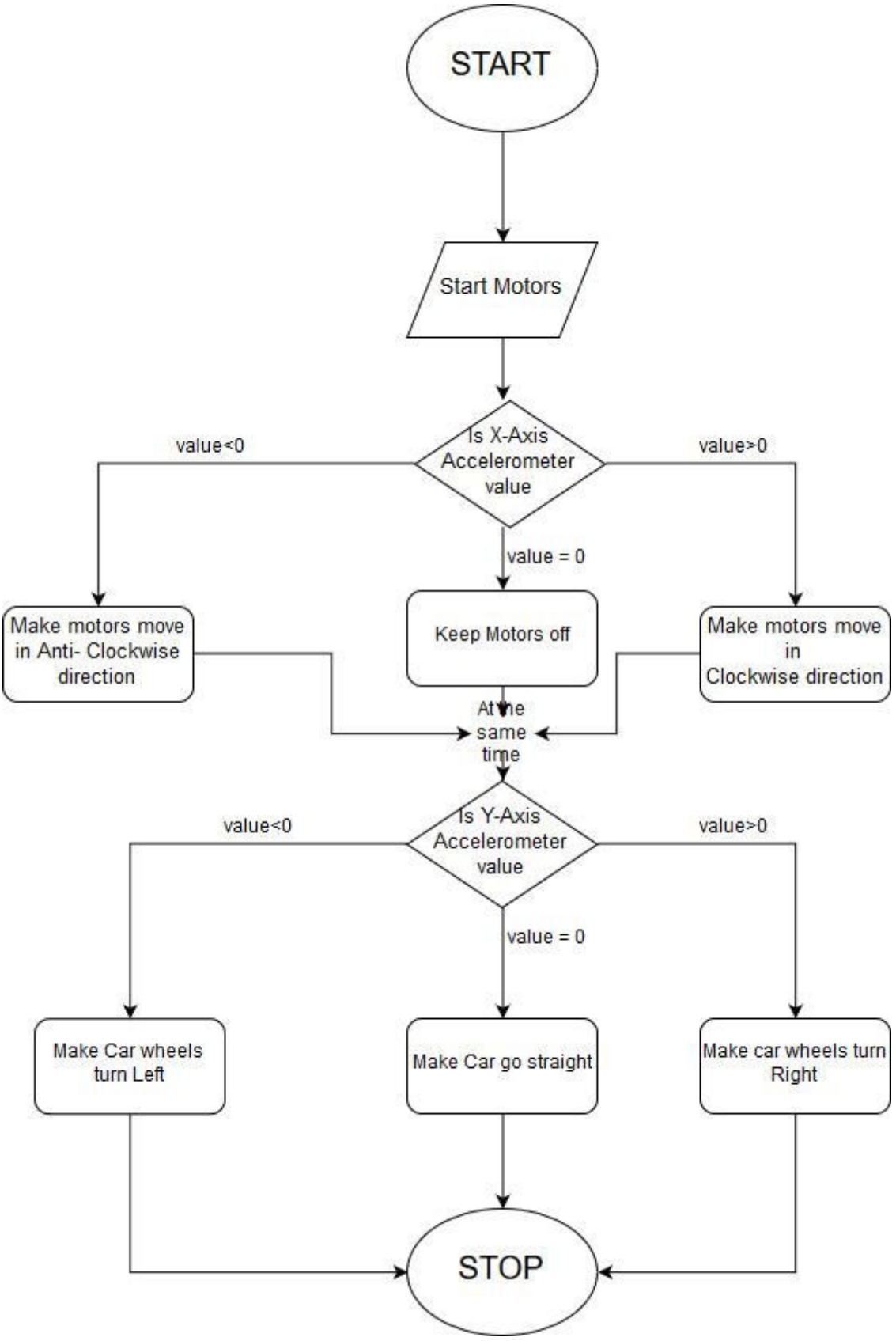
4) RF Module 433 MHz

5) 16X2 LCD

6) DC Motors

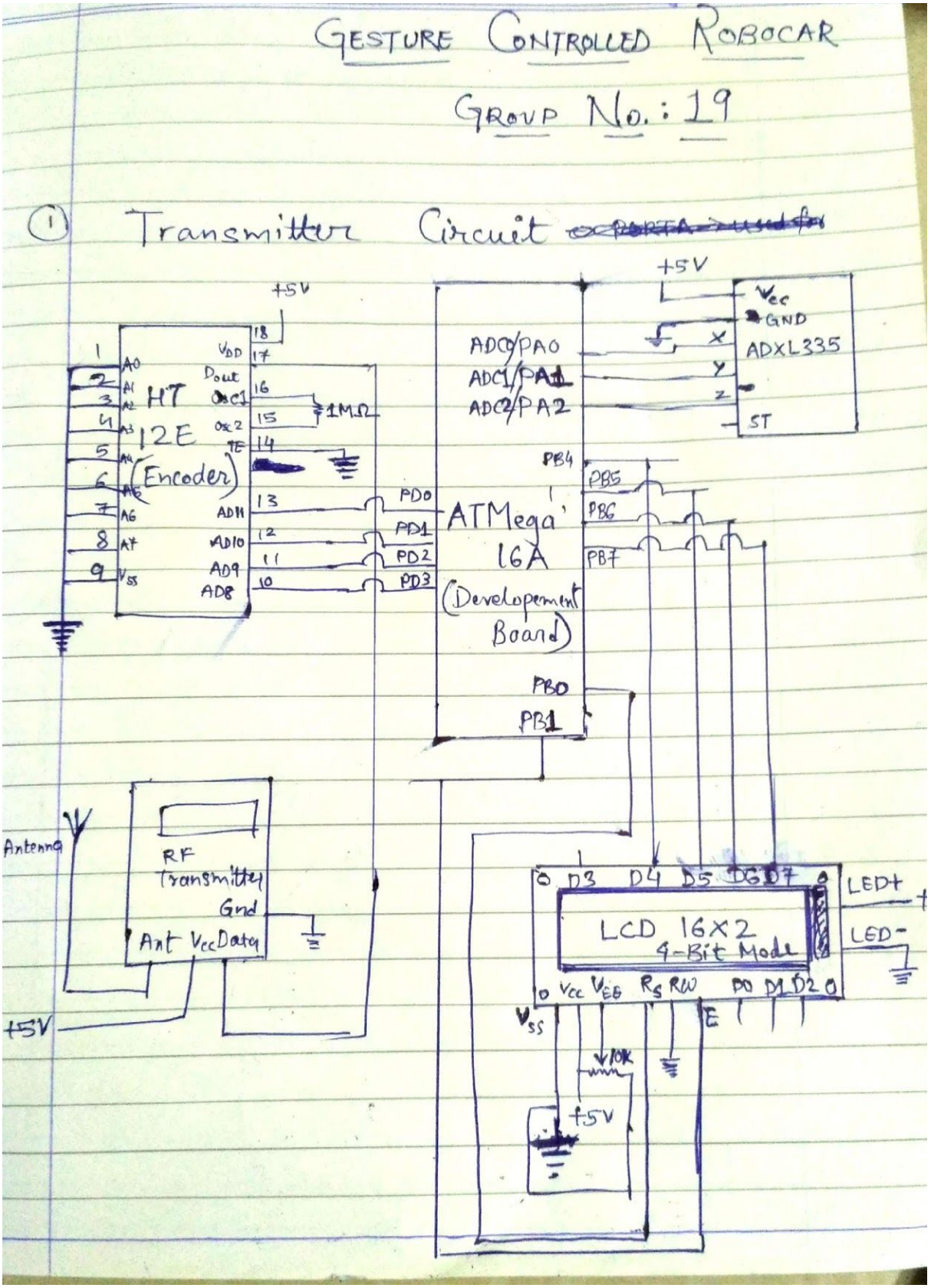
7) Car Chassis

Flow Chart:



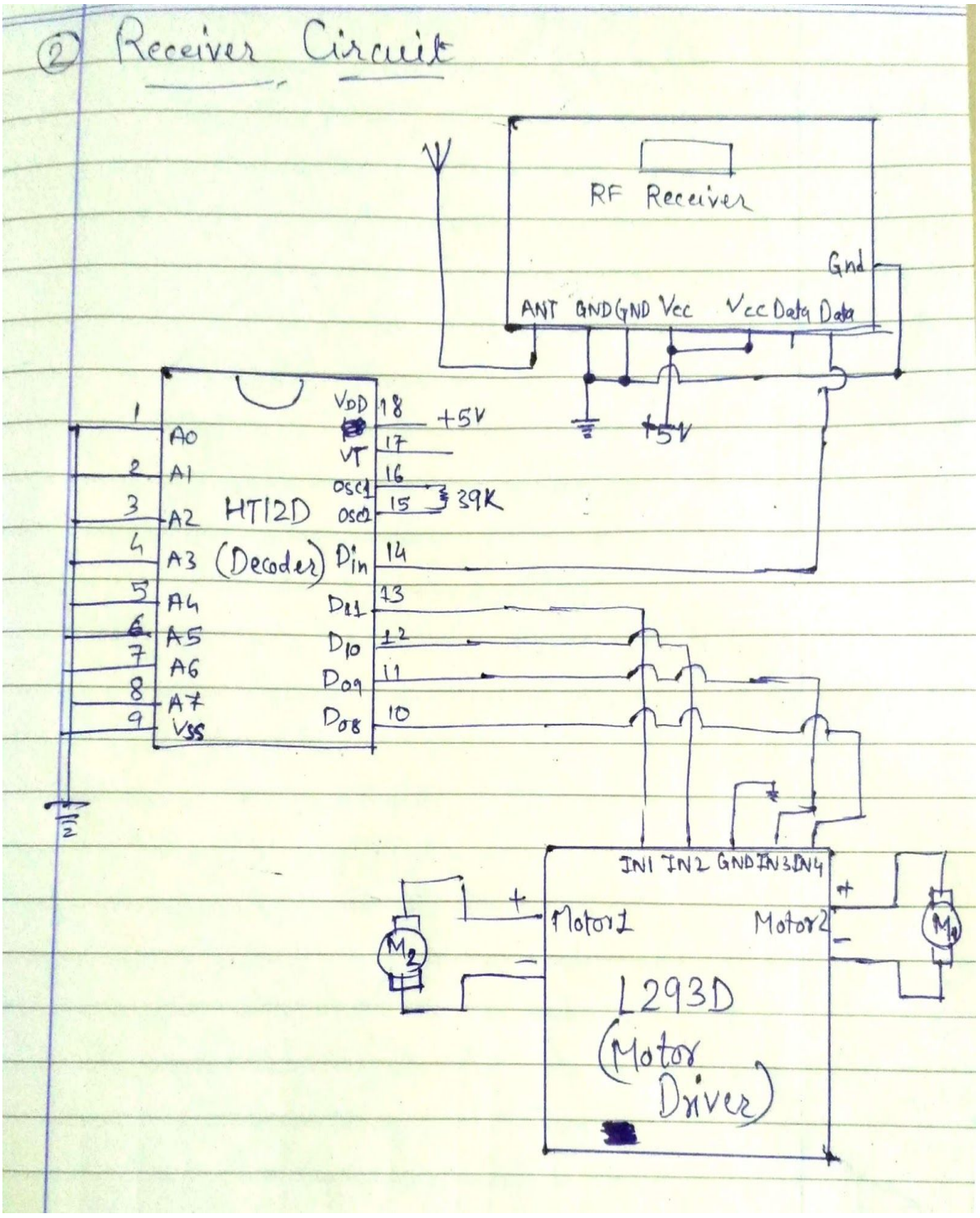
Working Circuit:-

1) Transmitter Circuit:

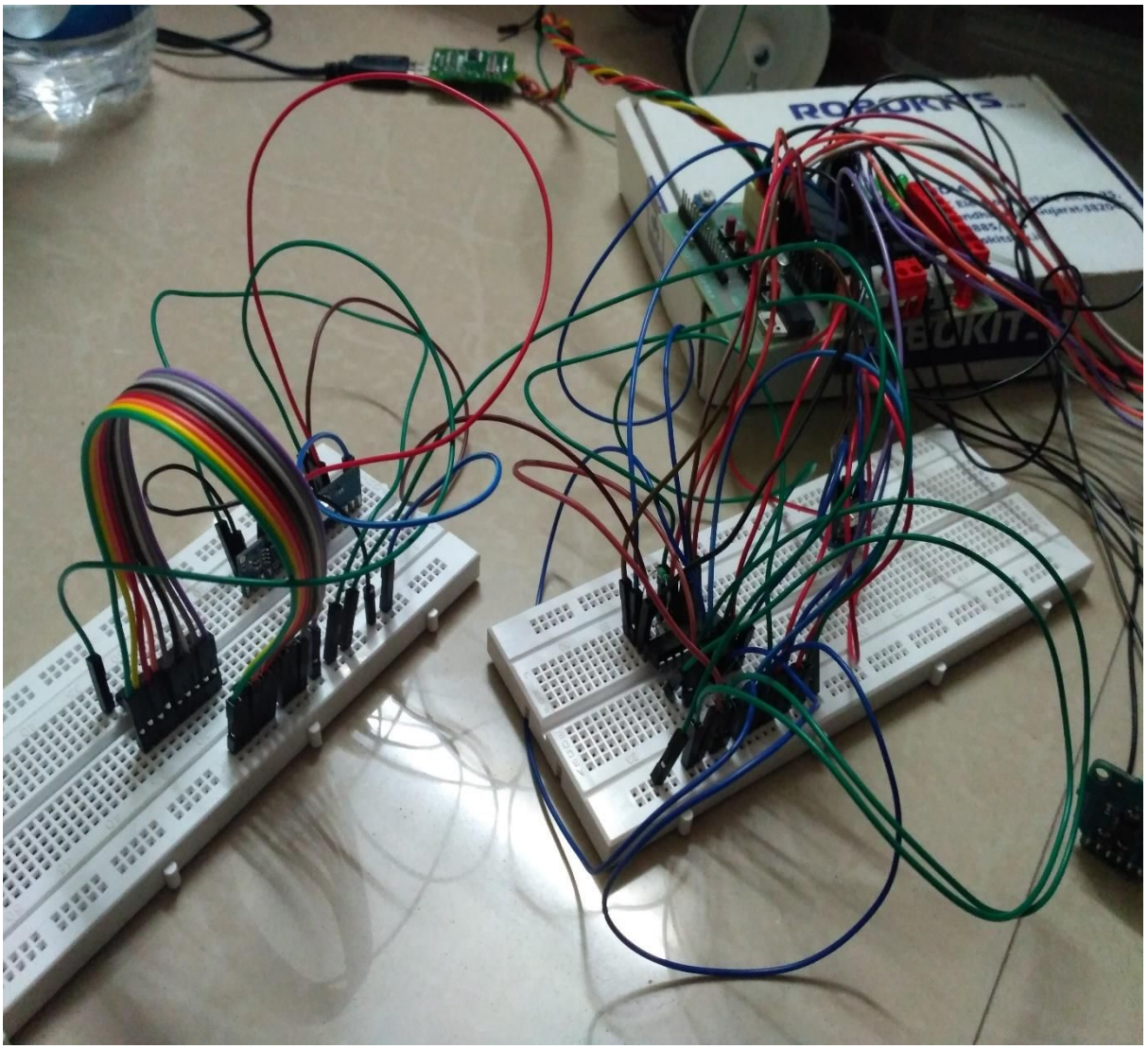


2) Receiver Circuit:

② Receiver Circuit



Circuit Photo:



Working Code:

```
#define F_CPU 8000000UL /* Define CPU clock Frequency e.g. here
its 8MHz */
#include <avr/io.h> /* Include AVR std. library file */
#include <util/delay.h> /* Include defined delay header file */
#include <stdlib.h> /* Include standard library file */

#define LCD_Dir DDRB /* Define LCD data port direction */
```

```
#define LCD_Port PORTB /* Define LCD data port */
#define RS PB0 /* Define Register Select */

#define EN PB1 /* Define Enable signal pin */
#define X_MIN 310
#define X_MAX 390
#define Y_MIN 310
#define Y_MAX 390

void LCD_Command( unsigned char cmnd )
{
LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0);
LCD_Port &= ~ (1<<RS);  /* RS=0, command reg. */
LCD_Port |= (1<<EN);  /* Enable pulse */
_delay_us(1);
LCD_Port &= ~ (1<<EN);

_delay_us(200);

LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);
LCD_Port |= (1<<EN);
_delay_us(1);
LCD_Port &= ~ (1<<EN);
_delay_ms(2);
}


void LCD_Char( unsigned char data )
{
LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0);
LCD_Port |= (1<<RS);  /* RS=1, data reg. */
LCD_Port|= (1<<EN);
_delay_us(1);
LCD_Port &= ~ (1<<EN);

_delay_us(200);

LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower
     nibble */
LCD_Port |= (1<<EN);
_delay_us(1);
LCD_Port &= ~ (1<<EN);
_delay_ms(2);
}

void LCD_String (char *str)    /* Send string to LCD function */
{
int i;
for(i=0;str[i]!=0;i++) /* Send each char of string */
{
           LCD_Char (str[i]);
```

```
}
}


void LCD_String_xy (char row, char pos, char *str)
{
if (row == 0 && pos<16)
          LCD_Command((pos & 0x0F)|0x80);
else if (row == 1 && pos<16)
          LCD_Command((pos & 0x0F)|0xC0);
LCD_String(str); /* Call LCD string function */
}

void LCD_Clear()
{
LCD_Command (0x01); /* clear display */
LCD_Command (0x80); /* cursor at home position */
}

void LCD_Init (void) /* LCD Initialize function */
{
_delay_ms(20); /* LCD Power ON delay always >15ms */
LCD_Dir = 0xFF;    /* Make LCD command port direction as o/p */
LCD_Command(0x02);     /*send for 4 bit initialization of LCD  */
LCD_Command(0x28);              /*use 2 line and initialize
                                      5*7 matrix in (4-bit mode)*/
LCD_Command(0x0c);              /*display on cursor off*/
LCD_Command(0x06);              /*increment cursor (shift cursor
     to right)*/
LCD_Command(0x01);              /*clear display screen*/
}

void LCD_Number(int number,unsigned char radix)
{
char *number_string="00000";
itoa(number,number_string,radix);
LCD_String(number_string);
}

void ADC_Init() /* ADC InitialiAzouttion function */
{
DDRA = 0x00; /* Make ADC port as input */
ADCSRA = 0x87; /* Enable ADC, with freq/128  */
ADMUX = 0x40; /* Vref: Avcc, ADC channel: 0 */
}

int ADC_Read(char channel)    /* ADC Read function */
{
ADMUX = 0x40 | (channel & 0x07);
ADCSRA |= (1<<ADSC); /* Start ADC conversion */
```

```
while (!(ADCSRA & (1<<ADIF))); /* Wait until end of conversion by
     polling ADC interrupt flag */
ADCSRA |= (1<<ADIF);    /* Clear interrupt flag */
_delay_ms(1);         /* Wait a little bit */
return ADCW;         /* Return ADC word */
}


int main(void)
{
int ADC_X_VALUE,ADC_Y_VALUE,ADC_Z_VALUE;
ADC_Init();             /* Initialize ADC */
LCD_Init(); /* initialization of LCD*/

DDRD=0xFF;
while(1)
{
ADC_X_VALUE = ADC_Read(0); /* Read X, Y, Z axis values */
ADC_Y_VALUE = ADC_Read(1);
ADC_Z_VALUE = ADC_Read(2);
LCD_Command(0x01);
/*Clear screen*/
LCD_String("Tilt your Hand:");
/*String display in 1st row of LCD*/
LCD_Command(0xc0);
/*Cursor moves to 2nd row 1st column of LCD*/
if(ADC_X_VALUE<X_MIN)
{
PORTD = 0x0A;
LCD_String("Moving Forward");
}
else if(ADC_X_VALUE>X_MAX)
{
PORTD = 0x05;
LCD_String("Moving Backward");
}
else if(ADC_Y_VALUE>Y_MAX)
{
PORTD = 0x08;
LCD_String("Moving Right");
}
else if(ADC_Y_VALUE<Y_MIN)
{
PORTD = 0x02;
LCD_String("Moving Left");
}
else
{
PORTD = 0X0F;
LCD_String("Stop");
}
```

```
}
}
```