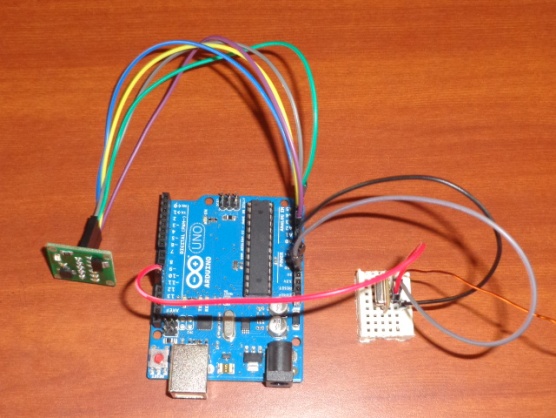
Gesture Controlled Robot



**ABSTRACT**

Gesture is a very natural human communication capability. Therefore, it should lend itself to easily learned interaction techniques. A distinguishing feature of the gesture communication channel is that it allows one to act on one’s environment as well as to retrieve information from it.

The objective of this project is to build a robot that can be controlled by gesture wirelessly. User is able to control motions of the robot by wearing the controller glove and performing predefined simple gestures. The lcd indicates the present status of the robot’s movement.

This project provides a basic platform for many potential applications such as wireless controlled car racing games, gesture human-machine, interfacing, etc.

For this project, ulk kit, avr, accelerometer are employed for the controlled bot. H-bridge, encoder, accelerometer sensor are employed for the controller. However, the hardware is also ready for XBee wireless protocol.

INTRODUCTION

Hand gesture recognition has been a very active research topic in recent years with motivating applications such as human computer interaction, robot control and sign language interpretation.

A gesture controlled robot is a kind of robot is a kind of robot which can be controlled by hand gestures and not by the old buttons. A small transmitting device needs to be worn on the hand which includes an accelerometer. This will transmit an appropriate command to the robot so that it can move as per the direction required. The transmitting device includes a comparator IC for analog to digital conversion. And an encoder IC (HT12E) is used to encode the 4 bit data and then it will transmit via RF module.

At the receiving end an RF Receiver module receives the encoded data which is decoded by a decoder IC (HT12D). This is then processed by the microcontroller and finally the motor driver to control the motors.

Working Principle-

The basic working principle for our robot is passage of the data signals of accelerometer readings to the AVR board fitted on the bot. The program compiled in that AVR runs according to that value, which make the bot function accordingly .While we have used dual-axis accelerometer. In which, one axis will control the movement in forward or backward direction and other axis will control the movement in left and right direction.One major rationale behind the project was to design a wireless car that steers and accelerates by means of an accelerometer instead of a joystick. The accelerometer would be fitted to the hand. The user can tilt his hand to left/right and forward/backward to control the direction and speed of the bot, respectively. The other use of the robot it can be used to store the commonly visited places i.e their direction to the destination from the given place. This can be used to

Move the bot with just the press of button,without the need of guesture.

**Block Diagram**

ANTENNA 2

ANTENNA 1

RECEIVER

TRANSMITTER

DECODER

ENCODER

MICROCONTROLLER

COMPARATOR

ACCELEROMETER

MOTOR DRIVER

**WIRELESS ROBOT**

[Cite your source here.]

**GESTURE TRANSMITTER**

[Cite your source here.]

**FLOWCHART**

START

Make gesture

Yes

No

No

Yes

Yes

END

Is the desired position attained?

Microcontroller processes data and makes suitable decision.

Signal transmitted through Xbee and received at the receivers end.

Compare with the previous position

And get the direction of movement.

If

Gesture

Made?

Instruction sent to motors.

**Description**

The robot mainly consists of two parts. The transmitting or gesture device and the receiver or the robot. In both antennae are required for reception and transmission purpose.

The gesture device has the following components-

1. Accelerometer- This is a sensor which gives analog data while moving in X, Y direction (depends on sensor type).There are generally 6 pins.

a) Vdd- connected to +5 V

b) GND-connected to ground for biasing.

c) X-We receive analog data for X direction movement on this pin. Similar pins for Y and Z direction movement is also present.

d) ST – This pin is used to set sensitivity of the accelerometer.

2. Comparator- Once the movement of the gestured section is sensed by accelerometer, the voltage levels corresponding to movement of the robot along X and Y axes are compared by a comparator. This simultaneously changes the analog voltage into digital value by comparing the analog voltage to a reference voltage and gives a high or low output. LM324 IC is used which is a 14 pin IC.

3. Encoder-The HT12E IC is a 4bit encoder which encodes the input data applied on it.

This is an eighteen pin IC.

4. RF Transmitter Module-Data is transmitted from the gesture transmitter through the RF transmitter module (TX).

5.EEEPROM-the inbuilt memory in the avr board.

The receiver or the bot has the following components-

1. RF Receiver Module-This will receive the data which is transferredfrom the gestured device. It also works similar to the transmitter module.

2. DECODER-The IC HT12D (18 pin) is a decoder that converts serial data into parallel which is received by the RF Receiver module. The input data is decoded when no errors are found. A valid transmission is indicated by a high signal at VT pin (pin 17).

3. Processor (AVR-atmega8)-Processing is the most important part of the robot. After getting the data from the decoder, we have to make the decisions based on those values. Here comes the microcontroller to give the device decision capability.

4. Actuators-These devices gives the movement or to do a task like motors. We need a motor driver for running the motors through the controllers.

We can also use Xbee (series1) which is a RF-module.

**CODE**:-

**RECIEVER**

#include <VirtualWire.h>

byte message[VW\_MAX\_MESSAGE\_LEN]; // a buffer to hold the incoming messages

byte msgLength = VW\_MAX\_MESSAGE\_LEN; // the size of the message

byte a[]={'r','l','b','f','s','t','x','y','m','z'};

void setup()

{

Serial.begin(9600);

Serial.println("Ready");

pinMode(8,OUTPUT);

pinMode(9,OUTPUT);

pinMode(7,OUTPUT);

pinMode(6,OUTPUT);

pinMode(13,OUTPUT);

pinMode(3,OUTPUT);

pinMode(4,OUTPUT);

// Initialize the IO and ISR

vw\_setup(2000); // Bits per sec

vw\_rx\_start(); // Start the receiver

}

void loop()

{

start:

if (vw\_get\_message(message, &msgLength)) // Non-blocking

{

one:Serial.print("Got: ");

for (int i = 0; i < msgLength; i++)

{

Serial.write(message[i]);

}

Serial.println();

if(message[0]==a[0]) //r

{

digitalWrite(8,LOW);

digitalWrite(9,LOW);

digitalWrite(7,HIGH);

digitalWrite(6,LOW);

digitalWrite(13,LOW);

}

if(message[0]==a[1])//l

{

digitalWrite(8,HIGH);

digitalWrite(9,LOW);

digitalWrite(7,LOW);

digitalWrite(6,LOW);

digitalWrite(13,LOW);

}

if(message[0]==a[2])//b

{

while(1)

{

digitalWrite(8,HIGH);

digitalWrite(9,LOW);

digitalWrite(7,HIGH);

digitalWrite(6,LOW);

digitalWrite(13,LOW);

if (vw\_get\_message(message, &msgLength))

{

if(message[0]!=a[2])

{

goto one;

}

}

}

}

if(message[0]==a[3])//f

{

while(1)

{

digitalWrite(8,LOW);

digitalWrite(9,HIGH);

digitalWrite(7,LOW);

digitalWrite(6,HIGH);

digitalWrite(13,LOW);

if (vw\_get\_message(message, &msgLength))

{

if(message[0]!=a[3])

{

goto one;

}

}

}

}

if(message[0]==a[4])

{

digitalWrite(8,LOW);

digitalWrite(9,LOW);

digitalWrite(7,LOW);

digitalWrite(6,LOW);

digitalWrite(13,LOW);

}

if(message[0]==a[5])

{

digitalWrite(3,HIGH);//light ON

//digitalWrite(4,LOW);// OFF

}

if(message[0]==a[6])

{

digitalWrite(4,HIGH);//light Off

//digitalWrite(3,LOW);// On buzzer

}

if(message[0]==a[7])

{

// digitalWrite(4,LOW);//light Off

digitalWrite(3,LOW);// OFF buzzer

}

if(message[0]==a[8])

{

// digitalWrite(4,LOW);//light Off

digitalWrite(4,LOW);// OFF buzzer

}

if(message[0]==a[9])

{

digitalWrite(4,LOW);

digitalWrite(3,LOW);

digitalWrite(8,LOW);

digitalWrite(9,LOW);

digitalWrite(7,LOW);

digitalWrite(6,LOW);

digitalWrite(13,LOW);

goto start;

}

}

}

**TRANSMITTWER**

const int groundpin = 18; // analog input pin 4 -- ground

const int powerpin = 19; // analog input pin 5 -- voltage

const int xpin = A3; // x-axis of the accelerometer

const int ypin = A2; // y-axis

const int zpin = A1;

char a[6];

byte b[100];

int i=0;

int j=0;

#include <VirtualWire.h>

#include<EEPROM.h>

void setup()

{

Serial.begin(9600);// Initialize the IO and ISR

vw\_setup(2000); // Bits per sec

pinMode(groundpin, OUTPUT);

pinMode(powerpin, OUTPUT);

digitalWrite(groundpin, LOW);

digitalWrite(powerpin, HIGH);

pinMode(6,OUTPUT);

pinMode(7,OUTPUT);

pinMode(4,INPUT);

}

void loop()

{

boolean n;

delay(100);

int x=analogRead(xpin);

int y=analogRead(ypin);

int z=analogRead(zpin);

x=map(x,300,500,0 ,15);

y=map(y,300,500,0 ,15);

z=map(z,300,500,0 ,15);

n=digitalRead(4);

if(n==false)

{

//itoa(x,a,10);

Serial.print(x);

Serial.print("\t");

Serial.print(y);

Serial.print("\t");

Serial.println(z);

delay(100);

if(x<4 && y>=5 && y<9 && z>3)

{digitalWrite(7, LOW);

digitalWrite(6, HIGH);

//send(a);

send("l");

b[i]='l';

EEPROM.write (j,b[i]);

i++;

j++;

//send("r");

delay(800);

}

if(x>=9 && y>=5 && y<9 && z>3)

{

//send(a);

send("r");

b[i]='r';

EEPROM.write (j,b[i]);

j++;

i++;

digitalWrite(6, LOW);

digitalWrite(7, LOW);

delay(800);

}

if(y<4 && x>=5 && x<9 && z>3)

{

//send(a);

send("f");

b[i]='f';

EEPROM.write (j,b[i]);

j++;

i++;

digitalWrite(6, HIGH);

digitalWrite(7, HIGH);

delay(800);

}

if(y>=9 && x>=5 && x<9 && z>3)

{

//send(a);

send("b");

b[i]='b';

EEPROM.write (j,b[i]);

j++;

i++;

digitalWrite(7, HIGH);

digitalWrite(6, LOW);

delay(800);

}

/\*if(y>3 && x>3 && z<2)

{

send("");

}\*/

if(y>3 && x>3 && z>8)

{

send("s");

b[i]='s';

EEPROM.write (j,b[i]);

j++;

i++;

delay(800);

}

else

{

j++;

i++;

delay(800);

}

/\*else

{

send("s");

delay(1000);

}\*/

}

else

{ send(" ");

delay(500);

for(int j=0;j<101;j++)

{

if(b[j]=='f')

{

send("f");

delay(300);

}

if(b[j]=='b')

{

send("b");

delay(300);

}

if(b[j]=='r')

{

send("r");

delay(300);

}

if(b[j]=='l')

{

send("l");

delay(300);

}

if(b[j]=='s')

{

send("s");

delay(300);

}

}

delay(500);

}

}

void send (char \*message)

{

vw\_send((uint8\_t \*)message, strlen(message));

vw\_wait\_tx(); // Wait until the whole message is gone

}