**Research Paper**

**on**

**PROJECT TITLE**

**HAND GESTURE CONTROLLED ROBOTIC CAR USING ARDUINO.**

***Submitted in partial fulfillment of the***

***Requirements for the award of the degree***

***of***

BACHELORS OF ENGINEERING

in

MECHATRONICS ENGINEERING

**Submitted By: - Faculty Supervisor: -**

**NAME AND UID**

**INDERPREET SINGH**

**NANDAN KUMAR 20BEM8010**

**FRANCK LEONEL 19BEM1058**

**PARTHA PRABIR 20BEM8002**

**SUNNY 19BEM1047**



**UNIVERSITY INSTITUTE OF ENGINEERING**

**CHANDIGARH UNIVERSITY**

**GHARUAN, MOHALI, PUNJAB, INDIA-140413**

**2021**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Content**  **Candidate Declaration**  **Acknowledgement**  **Highlights**  **Abstract** | **Page no.** |
| 1. | Introduction.............................................................................. | 4-5 |
| 2. | Design and Construction…………………………………….. | 5-8 |
| 3. | Software ……………….…………………………………….   1. Coding 2. Experimental Setup | 9-18 |
|  |  |  |
| 4. | Result and Discussion………………………………...……... | 18 |
| 5. | Conclusion…………………………………………………... | 19 |
| 6. | Future Scope….…………….……………………………….. | 19 |
| 7. | References…………………………………………………… | 19-20 |

**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI**

**CANDIDATE'S DECLARATION**

I “Franck Leonel” hereby declare that the work embodied in this research paper entitled **“HAND GESTURE CONTROLLED ROBOTIC CAR USING ARDUINO”** in partial fulfillment of requirements for the award of degree of B.E (MECHATRONICS ENGINEERING) at **CHANDIGARH UNIVERSITY GHARUAN, MOHALI.** The work which is being presented in this research paper submitted to **Department of Mechatronics Engineering** is an authentic record of bonafied piece of work.

Signature

(FULL NAME)

Nandan Kumar

Franck Leonel Foti Hamekong

Partha Prabir

Sunny

Supervisor Signature

(Name and Designation)

Mr. Inderpreet Singh

**ACKNOWLEDGEMENT**

It gives me proud privilege to complete this project work. This is the only page where I have the opportunity to express my emotions and gratitude from the core of my heart.

It is a great pleasure in expressing sincere and deep gratitude towards our guide for her valuable and firm suggestions, guidance and constant support throughout the completion of project named “Hand Gesture Controlled Robotic Car using Arduino”. I am thankful to Chandigarh University for providing us various resources infrastructure facilities.

I also offer my most sincere thanks to my team members and staff members of Mechatronics Department, University Institute of Engineering, Chandigarh University for cooperation provided by them in many ways.

**NAME**

**(Semester, Mechatronics Eng.)**

Nandan Kumar

Franck Leonel Foti Hamekong

Partha Prabir

Sunny

(Semester 5, Mechatronics Eng.)

**HIGHLIGHTS**

Arduino UNO, Accelerometer, Gesture control.

**ABSTRACT**

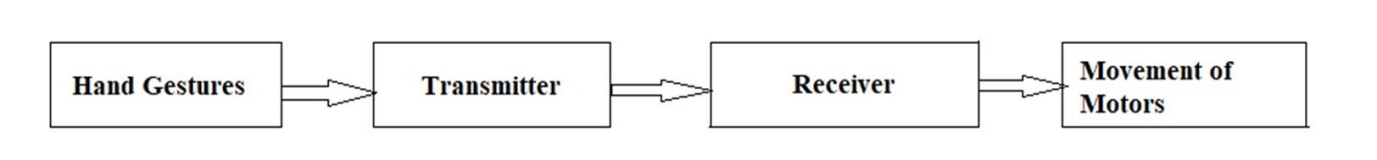
With the evolution of technology, engineers are seeking the most convenient way to complete different tasks with very little of human efforts. To do so, terms like automated systems, smart devices have come to light. Our project is an Arduino based robotic car that gets controls from hand gestures.

Here, the hand positioning is assigned different functions which in turn controls the car. The project mainly has two parts: the transmitter and receiver. The accelerometer on the transmitter circuit detects the position of the hand in which each movement generates some value corresponding to its axis. This value is sent to the receiver which decodes it and gives it to the motor driver which instructs the motors to move the car accordingly. In short, the car is controlled according to the signals it receives and are sent by the gestures of the hand.

1. **INTRODUCTION**

Hand Gestures can be regarded as a mode of communication while interacting with the machine robot by which we can eliminate the use of primary modes like Remote, joystick, etc. The hand gesture robot brings several benefits reduce human efforts and carry out effective results. Hand gesture robot uses the simple module like Arduino, accelerometer and HC-05.

Nowadays, there are a lot of research made on controlling of system or device with human Gestures, i.e., eye movement, facial expression or hand gestures which eliminates some barriers like language barrier for robot. Arduino UNO is used to interface different devices whereby it executes commands by interpreting the collection of data received from these interfaced devices. The Device uses the accelerometer to detect the any motion of the hand and transmits the data via a Bluetooth module to receiving end and performs certain task using Arduino micro controller.

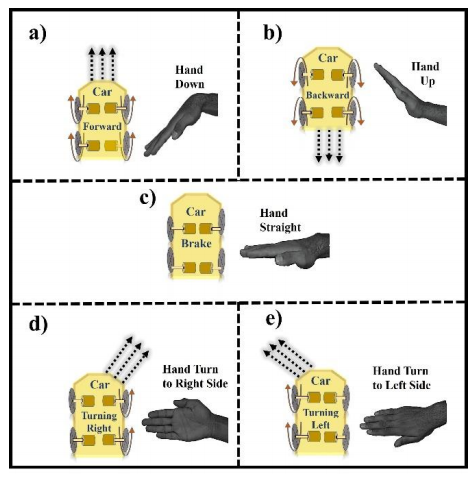


Sensors are used to detect the gestures of hand and to transmit the data, transmitter and receiver module is used. The exact movements are interpreted by Arduino Uno and Accelerometer. For the efficient motion of the robot, motors are connected in cross connection pattern to increase the turning efficiency of the automated car. Two different circuits namely transmitter & receiver circuits to establish a proper and secured wireless connection between a person and automated car.

As the name itself suggests, this is automated car controlled by hand gestures of person. For simple movements of robot car (forward, backward, left, right and in stopped position), if a person’s hand a tilted down then car moves forward direction, if tilted above then cars move in backward direction, if tilted to right then car turns in right direction and if tilted the left direction, then car turns in left direction. As mentioned, an accelerometer gyroscope is used. The sensors have 3 axis x, y, z which detect position of a person’s hand and gives an output in range of serial data. This data is received by the Arduino UNO micro-controller which is computed and then forwarded to the transmitter module. Two HC-05 Bluetooth modules are used; one on the transmitter circuit to encode and send the data and the other on the receiver circuit to receive the encoded data from the transmitter. Now at receiver end, this encoded data is captured by receiver module and then data is sent to decoder IC. This decoded data is then interpreted by the Arduino UNO and instructions are sent to the L298N motor driver to run the motors in the desired direction. All this process is described in a block diagram part.

1. **DESIGN AND CONSTRUCTION**

In this section of our paper, the different components used in our project are explained. As already mentioned, the project mainly is divided into the transmitter and receiver circuit. The parts of each circuit are given below.



**FOR TRANSMITTER**

* Arduino Uno
* HC-05 Bluetooth module
* MPU6050 Gyro with accelerometer
* Battery (9-12V)
* Hand gloves

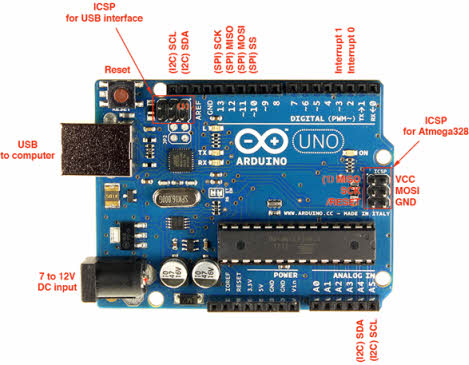
**FOR RECEIVER**

* Arduino Uno
* HC-05 Bluetooth module
* L298N H-Bridge motor driver
* Robot car chassis, DC motor and wheels
* Battery (9-12V)

***Arduino UNO***

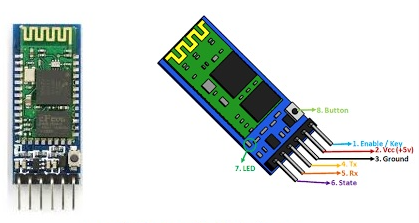
Arduino is an open-source electronics platform based on easy-to-use hardware and software which can support a microcontroller. This board can read inputs, light on a sensor, control a motor and many more applications. It is the brain of the car and is installed with some code. Instructions are sent to the devices connected to the board by use of the Arduino programming language or the Arduino IDE.

The board itself is made of different pins that serve various functions and can be used as of the wish of the programmer in order to complete any project.Simply connect it to the computer with a USB cable to get started with the Arduino board. It is flexible with hardware and software that are easy to use.

The Arduino UNO is a microcontroller board based on the Atmega328P with 14 digital I/O pins (Of which 6 are used as PMW output pins). It also has 6 analog input pins.

***HC-05***

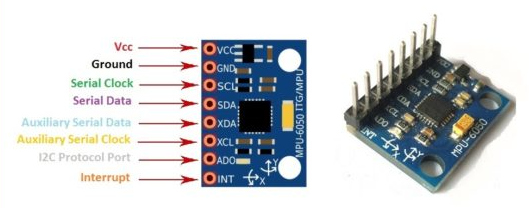
HC-05 Bluetooth Module is an easy-to-use Bluetooth SPP (Serial Port Protocol) module, that has been designed for transparent wireless serial connection setup. Its communication is via serial communication making it an easy way to interface with controller or PC. HC-05 Bluetooth module has the ability of switching mode between master and slave mode which means it able to use neither receiving nor transmitting data. It uses the 2.45GHz frequency band. The rate of transfer of the data can vary up to 1Mbps and is in range of 10 meters.



***MPU6050***

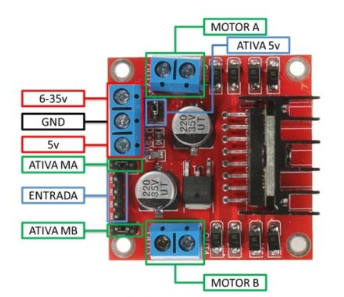
MPU6050 is a Micro Electro-Mechanical System (MEMS) consisting of three-axis accelerometer and three-axis gyroscope. With this device, we can measure velocity, orientation, acceleration, displacement and other motion like features.

It consists of a 16-bit analog to digital converter hardware which enables it to capture three-dimension motion at the same time.Here, we use 6 axis gyroscopes which implies a sensor with 6 degrees of freedom consisting of three axes to measure the rate of turn and another three axis to measures the acceleration.



***L298N***

L298N module is a high voltage, high current dual full-bridge motor driver module used for controlling DC motor and stepper motor. It enables the control of both the speed and rotation direction of two DC motors. This module consists of an L298 dual-channel H-bridge motor driver IC. This module uses two techniques for the control of the speed and rotation direction of the motors. These are PMW for controlling the speed and H-Bridge for controlling rotation direction. These modules can control two DC motors or one stepper motor at the same time.



***DC Motor***

DC motor converts direct current into mechanical motion which could be rotary or linear motion. DC motor works on the principle that when a current carrying conductor is placed in a magnetic field, the conductor experiences a mechanical force. The speed of a DC motor can be controlled by armature control or by speed control by changing the voltage applied to these. DC motors are used for the movement of the robotic car.



***Battery***

A battery consists of one or more electrochemical cells. It converts chemical energy to the electrical energy. The purpose of battery is to supply 9 -12 volts to operate DC motors.



1. **SOFTWARE**
2. **Coding**

The microcontroller is programmed such that when a certain range of tilting occurs, it will transmit signals to the motor to move to the direction required.

Arduino (IDE)software has been used as the interface between software and hardware of this project. The microcontroller needs a program to operate and execute the process associated with proposed design. The code is easily verified and compiled after it has been written.

Here, it connects to the Arduino hardware to upload programs. But before upload the program there is a need to select appropriate microcontroller like Arduino Uno from the Tool menu has been chosen. And for proper communication with computer and Arduino boards there is a need to select COM port from the tool menu.

**Transmitter Code**

#include "I2Cdev.h"

#include "MPU6050\_6Axis\_MotionApps20.h"

// Arduino Wire library is required if I2Cdev I2CDEV\_ARDUINO\_WIRE implementationis used in I2Cdev.h

#include "Wire.h"

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // CONNECT BT RX PIN TO ARDUINO 11 PIN | CONNECT BT TX PIN TO ARDUINO 10 PIN

#define OUTPUT\_READABLE\_YAWPITCHROLL

#define INTERRUPT\_PIN 2 // use pin 2 on Arduino Uno & most boards

#define LED\_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)

MPU6050 mpu;

bool blinkState = false;

// 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

VectorFloat gravity; // [x, y, z] gravity vector

float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector

float pitch = 0;

float roll = 0;

float yaw = 0;

int x;

int y;

// ================================================================

// === INTERRUPT DETECTION ROUTINE ===

// ================================================================

volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high

void dmpDataReady() {

mpuInterrupt = true;

}

// ================================================================

// === INITIAL SETUP ===

// ================================================================

void setup() {

// join I2C bus (I2Cdev library doesn't do this automatically)

#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

Wire.begin();

Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties

#elif I2CDEV\_IMPLEMENTATION == I2CDEV\_BUILTIN\_FASTWIRE

Fastwire::setup(400, true);

#endif

// initialize serial communication

// (115200 chosen because it is required for Teapot Demo output, but it's

// really up to you depending on your project)

Serial.begin(115200);

BTSerial.begin(38400); // HC-05 default speed in AT command more

while (!Serial); // wait for Leonardo enumeration, others continue immediately

// NOTE: 8MHz or slower host processors, like the Teensy @ 3.3V or Arduino

// Pro Mini running at 3.3V, cannot handle this baud rate reliably due to

// the baud timing being too misaligned with processor ticks. You must use

// 38400 or slower in these cases, or use some kind of external separate

// crystal solution for the UART timer.

// initialize device

Serial.println(F("Initializing I2C devices..."));

mpu.initialize();

pinMode(INTERRUPT\_PIN, INPUT);

// verify connection

Serial.println(F("Testing device connections..."));

Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));

// load and configure the DMP

Serial.println(F("Initializing DMP..."));

devStatus = mpu.dmpInitialize();

// supply your own gyro offsets here, scaled for min sensitivity

mpu.setXGyroOffset(126);

mpu.setYGyroOffset(57);

mpu.setZGyroOffset(-69);

mpu.setZAccelOffset(1869); // 1688 factory default for my test chip

// make sure it worked (returns 0 if so)

if (devStatus == 0) {

// turn on the DMP, now that it's ready

Serial.println(F("Enabling DMP..."));

mpu.setDMPEnabled(true);

// enable Arduino interrupt detection

Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));

attachInterrupt(digitalPinToInterrupt(INTERRUPT\_PIN), dmpDataReady, RISING);

mpuIntStatus = mpu.getIntStatus();

// set our DMP Ready flag so the main loop() function knows it's okay to use it

Serial.println(F("DMP ready! Waiting for first interrupt..."));

dmpReady = true;

// get expected DMP packet size for later comparison

packetSize = mpu.dmpGetFIFOPacketSize();

} else {

// ERROR!

// 1 = initial memory load failed

// 2 = DMP configuration updates failed

// (if it's going to break, usually the code will be 1)

Serial.print(F("DMP Initialization failed (code "));

Serial.print(devStatus);

Serial.println(F(")"));

}

// configure LED for output

pinMode(LED\_PIN, OUTPUT);

}

// ================================================================

// === MAIN PROGRAM LOOP ===

// ================================================================

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) {

// other program behavior stuff here

// .

// .

// .

// 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

// .

// .

// .

}

// reset interrupt flag and get INT\_STATUS byte

mpuInterrupt = false;

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!"));

// otherwise, check for DMP data ready interrupt (this should happen frequently)

} else if (mpuIntStatus & 0x02) {

// wait for correct available data length, should be a VERY short wait

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);

yaw = ypr[0] \* 180 / M\_PI;

pitch = ypr[1] \* 180 / M\_PI;

roll = ypr[2] \* 180 / M\_PI;

if (roll > -100 && roll < 100)

x = map (roll, -100, 100, 0, 100);

if (pitch > -100 && pitch < 100)

y = map (pitch, -100, 100, 100, 200);

Serial.print(x);

Serial.print("\t");

Serial.println(y);

if((x>=45 && x<=55) && (y>=145 && y <=155)){

BTSerial.write('S');

}else if(x>60){

BTSerial.write('R');

}else if(x<40){

BTSerial.write('L');

}else if(y>160){

BTSerial.write('B');

}else if(y<140){

BTSerial.write('F');

}

#endif

// blink LED to indicate activity

blinkState = !blinkState;

digitalWrite(LED\_PIN, blinkState);

}

}

**Receiver Code**

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(2, 3); //

char tiltDirection;

int motorInput1 = 11;

int motorInput2 = 10;

int motorInput3 = 9;

int motorInput4 = 8;

void setup() {

pinMode(motorInput1, OUTPUT);

pinMode(motorInput2, OUTPUT);

pinMode(motorInput3, OUTPUT);

pinMode(motorInput4, OUTPUT);

digitalWrite(motorInput1, LOW);

digitalWrite(motorInput2, LOW);

digitalWrite(motorInput3, LOW);

digitalWrite(motorInput4, LOW);

Serial.begin(115200); // Serial communication is activated at 38400 baud/s.

BTSerial.begin(38400); // HC-05 default speed in AT command more

}

void loop() {

if (BTSerial.available()) { // Witing for data incoming from the other XBee module

tiltDirection = BTSerial.read();

if(tiltDirection == 'F'){

Serial.println("Backward");

}else if(tiltDirection == 'B'){

Serial.println("Forward");

forward();

}else if(tiltDirection == 'R'){

Serial.println("Left");

left();

}else if(tiltDirection == 'L'){

Serial.println("Right");

right();

}else if(tiltDirection == 'S'){

Serial.println("Stop");

stopCar();

}

}

}

void forward()

{

/\*The pin numbers and high, low values might be different depending on your connections \*/

digitalWrite(motorInput1, LOW);

digitalWrite(motorInput2, HIGH);

digitalWrite(motorInput3, LOW);

digitalWrite(motorInput4, HIGH);

}

void reverse()

{

/\*The pin numbers and high, low values might be different depending on your connections \*/

digitalWrite(motorInput1, HIGH);

digitalWrite(motorInput2, LOW);

digitalWrite(motorInput3, HIGH);

digitalWrite(motorInput4, LOW);

}

void right()

{

/\*The pin numbers and high, low values might be different depending on your connections \*/

digitalWrite(motorInput1, LOW);

digitalWrite(motorInput2, HIGH);

digitalWrite(motorInput3, LOW);

digitalWrite(motorInput4, LOW);

}

void left()

{

/\*The pin numbers and high, low values might be different depending on your connections \*/

digitalWrite(motorInput1, LOW);

digitalWrite(motorInput2, LOW);

digitalWrite(motorInput3, LOW);

digitalWrite(motorInput4, HIGH);

}

void stopCar() {

digitalWrite(motorInput1, LOW);

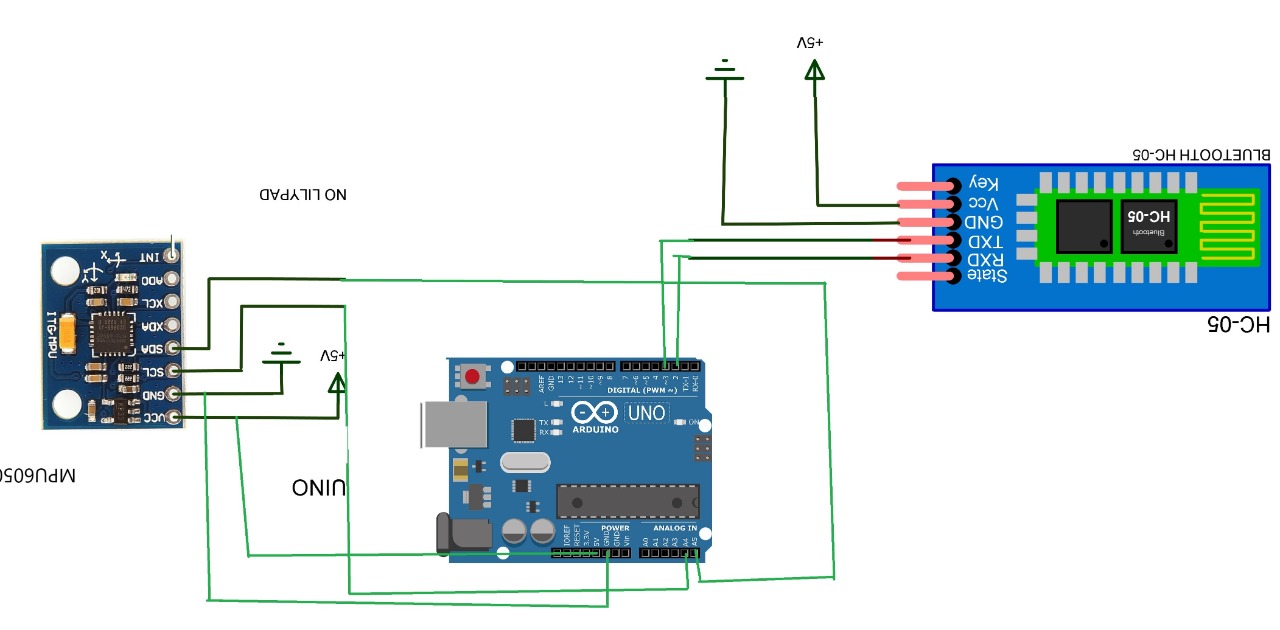
digitalWrite(motorInput2, LOW);

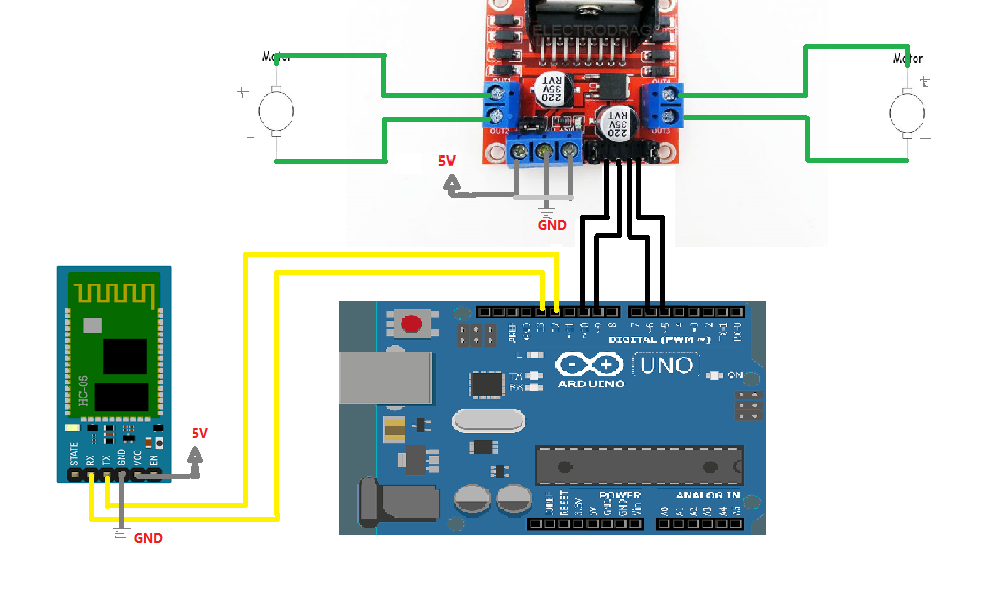
digitalWrite(motorInput3, LOW);

digitalWrite(motorInput4, LOW);

}

1. **Experimental Setup**

****

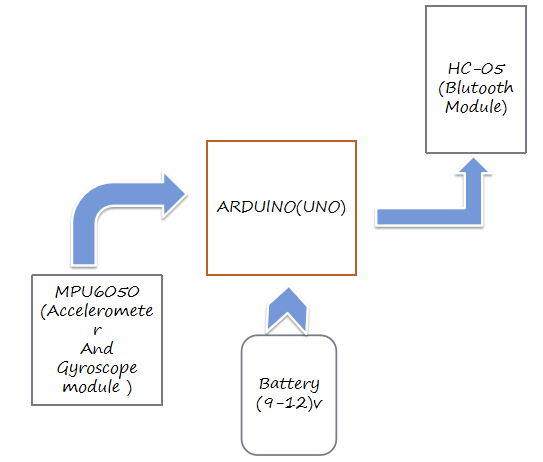


**Working of Transmitter**

In transmitter circuit we have an Arduino Uno with the accelerometer and with an encoder like our Bluetooth module.

Here first step is to recognize and detect the movements of hand by using Accelerometer MPU6050 and the position of hand is identified by using x and y scale. This data is supplied to Arduino Uno. Arduino has a micro controller ATMEGA 328 which can be programmed using Arduino (IDE) Software.

Following algorithm is used to detect and analyse the gestures through the accelerometer and in which direction automated robot should move.

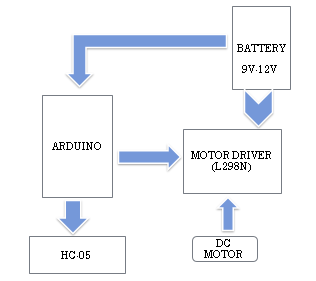


**Working of Receiver**

In receiver circuit, generally we use an Arduino with a Bluetooth module and the driver with the motor and some programming to control the motor and its direction. When the transmitter circuit gives the data in air then the receiver circuit captured those data and used our Bluetooth function as a decoder.

It decodes the data and sends four pin outputs of high or low digital signal to L298D motor driver board module.

And then the board runs the automated car in desired direction according to the gesture of the user’s hand.



1. **RESULT AND DISCUSSION**

As an outcome, we affirmed that our design could easily be implemented under real conditions at large-scale in the upcoming future, and it can be efficiently implemented in smart cars, personal home robots, quadrupedal robots, and hybrid robots etc.

Also, the human errors are reduced on a great scale and results are achieved with great accuracy. The limitations such as great power consumptions can be overcomed by use of strong batteries and low power consumption sensors. The human hand can move in various directions however only five of them are recognized by the car (forward, backward, left, right and in stopped position). So, the next step of this the project is to detect other hand gestures and getting the output from the car.

1. **CONCLUSION**

In this paper, a design for controlling a robot car based on Arduino Uno microcontroller

has been demonstrated, which can be programmed to respond to events based on hand gesture and to implement corresponding actions. The mechanism involves controlling the robot car based on hand gesture recognition (car moves similarly to the direction and position of hand) with the transmission of data done by HC-05 Bluetooth module. This project is beneficial where the humans are not able to perform a task but instead with some device using hand gestures.

1. **FUTURE SCOPE**

The concept of hand gesture robot may have applications in dangerous areas out of the reach of humans for the safety of workers whereby the robot is controlled at a distance, under the supervision of a human by the gestures of the hand. An example will be carrying heavy objects in industries. In this case, humans are replaced with hand gesture-controlled cars to perform the tasks. This technology enables large movements by just making small hand gestures.

The machines controlled by hand gestures will work efficiently in the field of defense, industry, medicine etc. In a hospital, a doctor can examine the patient with image manipulation or can move the tiny robot in the body using gestures and can find the problem.

Again, the concept of gesture control can help physically challenged people as they can move certain objects with less physical movement.

1. **REFERENCES**

[1] Yanan Chang, “Dynamic gesture recognition based on HMM,” MPhil.

thesis, South China University of Technology, Guangzhou, GD, China,

2012.

[2] Hang Zhao, “Motion control design of gesture recognition for mobile

robot,” M.S. thesis, University of Electronic Science and Technology,

Chengdu, SC, China, 2017.

[3] Chen Gao, “Research on the control of mobile robots based on dynamic

and static gestures,” M.S. thesis, Beijing University of Chemical

Technology, Nanjing, JS, China, 2017.

[4] Gould S J J, Brumby D P, Cox A L, et al., “Methods for HumanComputer Interaction Research,” in The, ACM Conference Extended.

ACM, 2015, pp. 2473-2474.

[5] Tianxing Wang, “Gesture-based remote control technology for

unmanned platforms,” M.S. thesis, Nanjing University, JS, China, 2017.

[6] V. Bonato, A. K. Sanches, Fernandes M M, et al., “A Real Time Gesture

Recognition System for Mobile Robots,” in Icinco 2004, Proceedings of

the First International Conference onInformatics in Control, Automation

and Robotics, Setubal, Portugal, August. DBLP, 2004, pp. 207-214.

[7] RM. Sanso, D. Thalmann, “A Hand Control and Automatic Grasping

System for Synthetic Actors,” Computer Graphics Forum, vol. 13, no. 3,

pp. 167-177, Aug. 1994.

[8] Robert Y. Wang, “Real-time hand-traking with a color glove,” in ACM

SIGGRAPH.ACM, 2009, pp. 1-8.

[9] Poularakis S, Katsavounidis I, “Finger detection and hand posture recognition based on depth information,” in IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE,2014, pp. 4329-4333. [10] Yunde Zhang, “Research and application of gesture recognition technology,” M.S. thesis, Anhui University, Hefei, AH, China, 2013. [11] Yuanxin Zhu, Guangyou Xu, “Appearance-based dynamic isolated gesture recognition,” Journal of Software, vol. 11, no.1, pp. 54-61, Jan.2000 [12] Xueping Zhao, Jianming Wang, Haiting Liu, et al., “Research on Recognition System of Indoor Robot Dynamic Gesture Command,” Computer Engineering and Applications, vol. 47, no. 33, pp. 209-212, Nov. 2011.