**Hand Gesture Control Car**

This is about how to make a gesture-controlled car by yourself. Basically this is a simple application of MPU-6050 3-axis Gyroscope, Accelerometer. You can do many more things. by understanding how to use it, how to interface it with Arduino and how to transfer its data over the Bluetooth modules. in this writeup, I will be focusing on Bluetooth to Bluetooth communication, in between two HC-05 Bluetooth modules.

**Components:**

Arduino uno

Arduino nano

L293d motor shield

Hc 05 blueoth module (x2)

9V battery (x2)

MPU 6050 gyro sensor

tt gear motor with tire (x2)

female to female wire

male to male wire

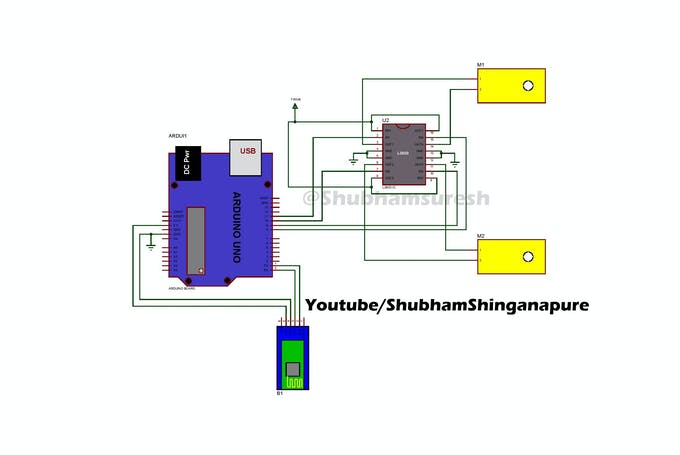
solid wire

solidering iron

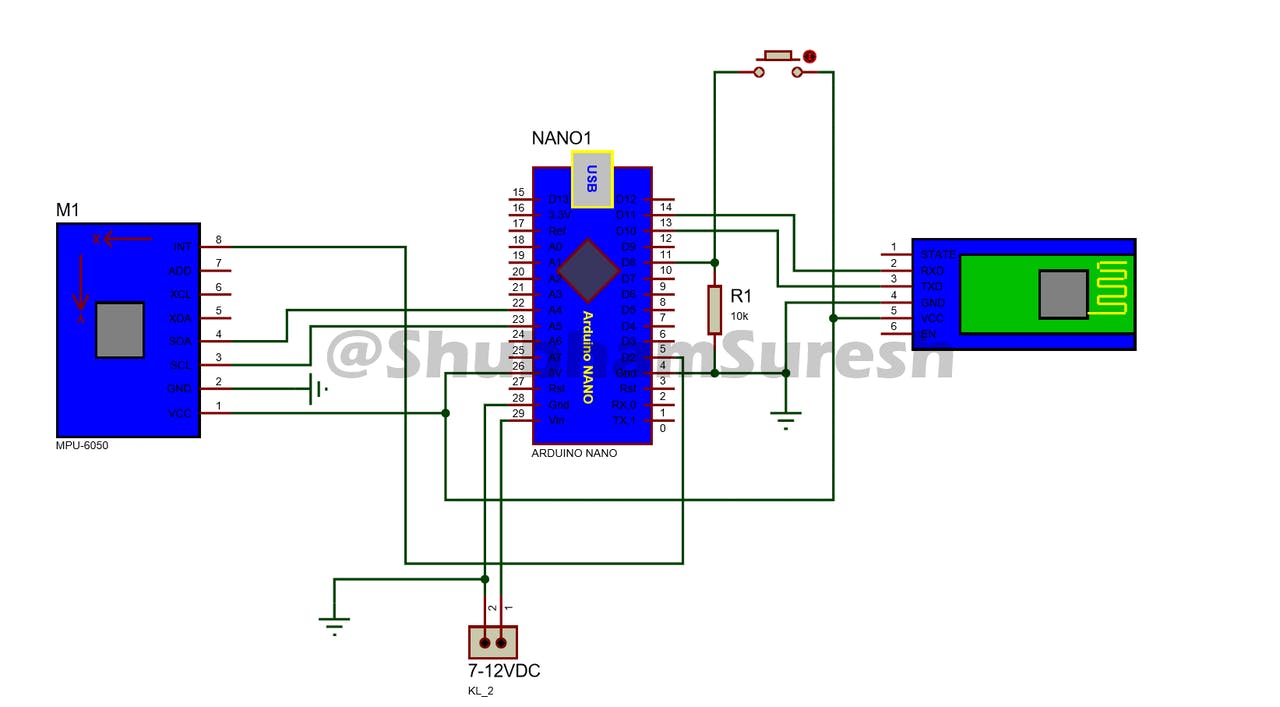
**Note: contact hallroad.org for all the components**

**Circuit Diagram:**

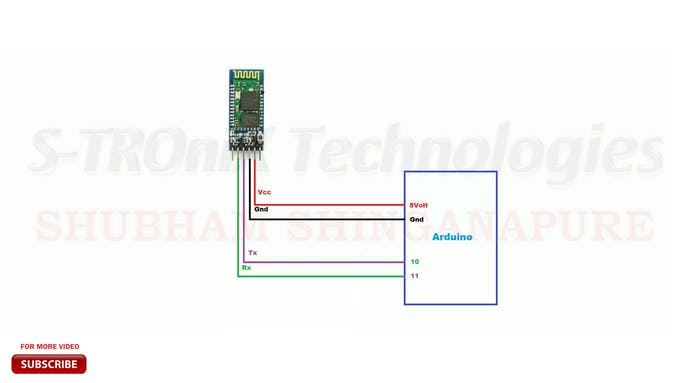
**Receiver:**



**Sender:**



**Programming:**



**Connect accrding to diargram and upload code bellow**

#include <SoftwareSerial.h>  
SoftwareSerial BTSerial(10, 11); // RX | TX  
void setup()  
{  
Serial.begin(9600);  
Serial.println("Enter AT commands:");  
BTSerial.begin(38400); // HC-05 default speed in AT command more  
}  
void loop()  
{  
// Keep reading from HC-05 and send to Arduino Serial Monitor  
if (BTSerial.available())  
Serial.write(BTSerial.read());  
// Keep reading from Arduino Serial Monitor and send to HC-05  
if (Serial.available())  
BTSerial.write(Serial.read());

**Now write this command for slave(reciever) bluetoth module on serial monitor**

AT

AT+UART=38400, 0, 0

AT+ROLE=0

AT+ADDR?

Note: remember or copy the address shown on serial monitor

**Now write this command for master(sender) bluetoth module on serial monitor**

AT

AT+UART=38400, 0, 0

AT+ROLE=1

AT+CMOD=0

AT+BIND= upper address which you get after AT+ADDR? Command

**Now disconnect Bluetooth module and upload that code in arduino uno by selecting correct port**

#include <AFMotor.h>

//initial motors pin

AF\_DCMotor motor1(1, MOTOR12\_1KHZ);

AF\_DCMotor motor2(2, MOTOR12\_1KHZ);

AF\_DCMotor motor3(3, MOTOR34\_1KHZ);

AF\_DCMotor motor4(4, MOTOR34\_1KHZ);

char command;

void setup()

{

Serial.begin(38400); //Set the baud rate to your Bluetooth module.

}

void loop(){

if(Serial.available() > 0){

command = Serial.read();

Stop();

//initialize with motors stoped

//Change pin mode only if new command is different from previous.

Serial.println(command);

switch(command){

case 'F':

forward();

break;

case 'B':

back();

break;

case 'L':

left();

break;

case 'R':

right();

break;

}

}

}

void forward()

{

motor1.setSpeed(255); //Define maximum velocity

motor1.run(FORWARD); //rotate the motor clockwise

motor2.setSpeed(255); //Define maximum velocity

motor2.run(FORWARD); //rotate the motor clockwise

motor3.setSpeed(255);//Define maximum velocity

motor3.run(FORWARD); //rotate the motor clockwise

motor4.setSpeed(255);//Define maximum velocity

motor4.run(FORWARD); //rotate the motor clockwise

}

void back()

{

motor1.setSpeed(255); //Define maximum velocity

motor1.run(BACKWARD); //rotate the motor anti-clockwise

motor2.setSpeed(255); //Define maximum velocity

motor2.run(BACKWARD); //rotate the motor anti-clockwise

motor3.setSpeed(255); //Define maximum velocity

motor3.run(BACKWARD); //rotate the motor anti-clockwise

motor4.setSpeed(255); //Define maximum velocity

motor4.run(BACKWARD); //rotate the motor anti-clockwise

}

void left()

{

motor1.setSpeed(255); //Define maximum velocity

motor1.run(BACKWARD); //rotate the motor anti-clockwise

motor2.setSpeed(255); //Define maximum velocity

motor2.run(BACKWARD); //rotate the motor anti-clockwise

motor3.setSpeed(255); //Define maximum velocity

motor3.run(FORWARD); //rotate the motor clockwise

motor4.setSpeed(255); //Define maximum velocity

motor4.run(FORWARD); //rotate the motor clockwise

}

void right()

{

motor1.setSpeed(255); //Define maximum velocity

motor1.run(FORWARD); //rotate the motor clockwise

motor2.setSpeed(255); //Define maximum velocity

motor2.run(FORWARD); //rotate the motor clockwise

motor3.setSpeed(255); //Define maximum velocity

motor3.run(BACKWARD); //rotate the motor anti-clockwise

motor4.setSpeed(255); //Define maximum velocity

motor4.run(BACKWARD); //rotate the motor anti-clockwise

}

void Stop()

{

motor1.setSpeed(0); //Define minimum velocity

motor1.run(RELEASE); //stop the motor when release the button

motor2.setSpeed(0); //Define minimum velocity

motor2.run(RELEASE); //rotate the motor clockwise

motor3.setSpeed(0); //Define minimum velocity

motor3.run(RELEASE); //stop the motor when release the button

motor4.setSpeed(0); //Define minimum velocity

motor4.run(RELEASE); //stop the motor when release the button

}

**Now disconnect Bluetooth module and upload that code in arduino nano by selecting correct port**

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files

// for both classes must be in the include path of your project

#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);

}

}

Note: mail me or contact me for libraries if you dont find

**Your device is ready best of luck and enjoy**