**Transmitter**

// include libraries

#include <Arduino.h>

#include <Wire.h>

#include "I2Cdev.h"

#include "MPU6050.h"

#include <RH\_ASK.h>

#ifdef RH\_HAVE\_HARDWARE\_SPI

#include <SPI.h>

#endif

// const

#define EMERGENCY\_STRING "EMERGENCY"

// Pin Conf

#define radioTransmitterPin 12

#define radioTransmissionSpeed 2000

#define emergencyBtnPin 4

// classes

MPU6050 accelerometer;

RH\_ASK radioTransmitter(radioTransmissionSpeed, radioTransmitterPin, radioTransmitterPin, 0);

struct SendData

{

/\* data \*/

int x;

int y;

};

SendData sendData;

// Define Values

#define MAX\_AX 17000

#define MIN\_AX -17000

#define MAX\_AY 17000

#define MIN\_AY -17000

#define DEBUG\_DELAY 100

// Variables

int16\_t accX, accY, accZ;

int16\_t gyroX, gyroY, gyroZ;

String commandString = "";

long long debugOldTime = 0;

// functions

void displayData();

void transmitData(String);

void checkEmergencyBtn();

void setup()

{

Serial.begin(9600);

// make sure I2C bus run

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

Wire.begin();

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

Fastwire::setup(400, true);

#endif

// initialization

accelerometer.initialize();

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

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

if(!radioTransmitter.init()){

Serial.println("Radio init failed");

}

// pinmode

pinMode(emergencyBtnPin, INPUT\_PULLUP);

}

void loop()

{

if(millis()-debugOldTime >= DEBUG\_DELAY){

checkEmergencyBtn();

// read accelerometer data

accelerometer.getMotion6(&accX, &accY, &accZ, &gyroX, &gyroY, &gyroZ);

sendData.x = map(accX, MIN\_AX, MAX\_AX, -255, 255);

sendData.y = map(accY, MIN\_AY, MAX\_AY, -255, 255);

displayData();

Serial.print("X=");

Serial.print(sendData.x);

Serial.print(" Y=");

Serial.println(sendData.y);

debugOldTime = millis();

checkEmergencyBtn();

// calculate

commandString = "x="+ String(sendData.x) + ";y="+ String(sendData.y)+ "; ";

Serial.println(commandString);

transmitData(commandString);

}

checkEmergencyBtn();

delay(2);

}

void transmitData (String data){

radioTransmitter.send((uint8\_t \*)data.c\_str(), commandString.length());

radioTransmitter.waitPacketSent();

}

void checkEmergencyBtn(){

bool state = digitalRead(emergencyBtnPin);

if(state == 0){

Serial.println(EMERGENCY\_STRING);

transmitData(EMERGENCY\_STRING);

delay(200);

}

}

void displayData (){

Serial.print("ax: "); Serial.print(accX);

Serial.print("; ay: "); Serial.print(accY);

Serial.print("; az: "); Serial.print(accZ);

Serial.print("; gx: "); Serial.print(gyroX);

Serial.print("; gy: "); Serial.print(gyroY);

Serial.print("; gz: "); Serial.print(gyroZ);

Serial.println();

}

**Wheel Chair Controller**

// headers

#include <Arduino.h>

#include <SoftwareSerial.h>

#include <RH\_ASK.h>

#ifdef RH\_HAVE\_HARDWARE\_SPI

#include <SPI.h>

#endif

#include <NewPing.h>

// define

#define radioSpeed 2000

#define radioRcvPin 2

#define RADIO\_MAX\_MSG\_LEN 12

#define NODEMCU\_RX\_PIN 10

#define NODEMCU\_TX\_PIN 9

#define GPS\_RX\_PIN 11

#define GPS\_TX\_PIN 12

#define DEBUG\_SERIAL\_BAUD 9600

#define NODEMCU\_SERIAL\_BAUD 9600

#define GPS\_SERIAL\_BAUD 4800

#define LEFT\_MOTOR\_1 A3

#define LEFT\_MOTOR\_2 A4

#define LEFT\_MOTOR\_SPEED 6

#define RIGHT\_MOTOR\_1 A0

#define RIGHT\_MOTOR\_2 A1

#define RIGHT\_MOTOR\_SPEED 5

#define FORWARD\_COMMAND\_SPEED\_MIN 50

#define TURN\_COMMAND\_SPEED\_MIN 50

#define MOTION\_DELAY 100

#define SONAR\_TRIG\_PIN 8

#define SONAR\_ECHO\_PIN 7

#define MAX\_ACCEPTED\_OBSTACLE\_DISTANCE 30.0 //cm

#define EMERGENCY\_STRING "EMERGENCY"

// class

RH\_ASK radioRev(radioSpeed, radioRcvPin, radioRcvPin, 0);

SoftwareSerial NodeMcu(NODEMCU\_RX\_PIN, NODEMCU\_TX\_PIN);

// SoftwareSerial GPSSerial(GPS\_RX\_PIN, GPS\_TX\_PIN);

// TinyGPSPlus gps;

NewPing ObstacleDistance(SONAR\_TRIG\_PIN, SONAR\_ECHO\_PIN);

// variables

uint8\_t radioBuff[RADIO\_MAX\_MSG\_LEN];

uint8\_t radioBuffLen = sizeof(radioBuff);

String radioData = "";

String commandData = "";

long long timeStamp = 0;

bool forwardCommand = false;

// functions

String readRadioCommand();

String getLocationData();

void sendSMS(String);

String getSplitValue(String, char, int);

void go(int, int);

void run();

void setup()

{

// put your setup code here, to run once:

Serial.begin(9600);

NodeMcu.begin(NODEMCU\_SERIAL\_BAUD);

if (!radioRev.init())

{

Serial.println("radio init failed");

}

Serial.println("starting");

pinMode(RIGHT\_MOTOR\_1, OUTPUT);

pinMode(RIGHT\_MOTOR\_2, OUTPUT);

pinMode(RIGHT\_MOTOR\_SPEED, OUTPUT);

pinMode(LEFT\_MOTOR\_1, OUTPUT);

pinMode(LEFT\_MOTOR\_2, OUTPUT);

pinMode(LEFT\_MOTOR\_SPEED, OUTPUT);

}

void loop()

{

run();

}

void run()

{

commandData = readRadioCommand();

// Serial.println(commandData);

if (commandData.length())

{

if (commandData.indexOf(EMERGENCY\_STRING) != -1)

{

// do emergency button works

Serial.println("EMMMMEEE");

sendSMS(getLocationData());

}

else

{

// locomotion work

// data format "x=255;y=255" x = formward, y means turn (positive = right turn, negative = left turn)

// if (millis() - timeStamp >= MOTION\_DELAY)

// {

Serial.println(commandData);

// extract data

String xData = getSplitValue(commandData, ';', 0);

String yData = getSplitValue(commandData, ';', 1);

int forwardMotion = getSplitValue(xData, '=', 1).toInt();

int turnMotion = getSplitValue(yData, '=', 1).toInt();

// accept tolerence

if (abs(forwardMotion) < FORWARD\_COMMAND\_SPEED\_MIN)

forwardMotion = 0;

if (abs(turnMotion) < TURN\_COMMAND\_SPEED\_MIN)

turnMotion = 0;

int leftMotorSpeed = forwardMotion - turnMotion;

int rightMotorSpeed = forwardMotion + turnMotion;

Serial.print(leftMotorSpeed);

Serial.print(" ");

Serial.println(rightMotorSpeed);

if(leftMotorSpeed > 0 && rightMotorSpeed >0){

forwardCommand = true;

}

else forwardCommand = false;

go(leftMotorSpeed, rightMotorSpeed);

// timeStamp = millis();

// }

}

// Serial.println(obstacleDistance);

if (forwardCommand )

{

float obstacleDistance = ObstacleDistance.ping\_cm();

if (obstacleDistance <= MAX\_ACCEPTED\_OBSTACLE\_DISTANCE && obstacleDistance >0)

go(0, 0);

}

}

}

String readRadioCommand()

{

// put your main code here, to run repeatedly:

if (radioRev.recv(radioBuff, &radioBuffLen))

{

// Serial.println("recv");

int i;

// radioRev.printBuffer("Got:", radioBuff, radioBuffLen);

radioData = "";

for (int j = 0; j < sizeof(radioBuff); j++)

{

radioData += (char)radioBuff[j];

}

// Serial.println(radioData);

// Serial.println(sizeof(radioBuff));

return String(radioData);

}

return String("");

}

void sendSMS(String location)

{

NodeMcu.println(location);

}

String getLocationData(){

GPSSerial.listen();

if(GPSSerial.available()>0){

Serial.print(GPSSerial.read());

if(gps.encode(GPSSerial.read())){

Serial.print(F("Location: "));

if (gps.location.isValid())

{

Serial.print(gps.location.lat(), 6);

Serial.print(F(","));

Serial.print(gps.location.lng(), 6);

return String(gps.location.lat(),6)+String(",")+String(gps.location.lng(),6);

}

}

return "";

}

return "";

}

String getSplitValue(String data, char separator, int index)

{

int found = 0;

int strIndex[] = {0, -1};

int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++)

{

if (data.charAt(i) == separator || i == maxIndex)

{

found++;

strIndex[0] = strIndex[1] + 1;

strIndex[1] = (i == maxIndex) ? i + 1 : i;

}

}

return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

}

void go(int leftSpeed, int rightSpeed)

{

bool leftForward = true;

bool rightForward = true;

if (leftSpeed < 0)

{

leftSpeed \*= -1;

leftForward = false;

}

if (rightSpeed < 0)

{

rightSpeed \*= -1;

rightForward = false;

}

if(leftSpeed > 255) leftSpeed = 255;

if (rightSpeed > 255) rightSpeed = 255;

analogWrite(LEFT\_MOTOR\_SPEED, leftSpeed);

digitalWrite(LEFT\_MOTOR\_1, leftForward);

digitalWrite(LEFT\_MOTOR\_2, !leftForward);

analogWrite(RIGHT\_MOTOR\_SPEED, rightSpeed);

digitalWrite(RIGHT\_MOTOR\_1, rightForward);

digitalWrite(RIGHT\_MOTOR\_2, !rightForward);

}