#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QTimer>

#include <QDebug>

#include <printf.h>

#include <stdlib.h>

#include <wiringPi.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <QMessageBox>

#include <QThread>

#include <QByteArray>

#include <printf.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/ioctl.h>

#include <QtSql/QSqlError>

#include <QtSql/QtSql>

#include <QtSql/QSqlQuery>

#include <QUrl>

#define MAXTIMINGS 85

#define DHTPIN 7

int dht11\_dat[5] = { 0, 0, 0, 0, 0 };

char buf[4];

int t;

int levelval;

int soilmotor;

QByteArray leveldata;

class SleeperThread : public QThread

{

public:

static void msleep(unsigned long msecs)

{

QThread::msleep(msecs);

}

};

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

// wiringPiSetup();

mcp3208 = new adc;

setup\_hcsr04(28,29);

// ROBO

pinMode(0,OUTPUT); // GPIO17

pinMode(1,OUTPUT); // GPIO18

pinMode(2,OUTPUT); // GPIO27

pinMode(3,OUTPUT); // GPIO22

digitalWrite(0,0);

digitalWrite(1,0);

digitalWrite(2,0);

digitalWrite(3,0);

// l293d for servo

pinMode(4,OUTPUT);

pinMode(5,OUTPUT);

digitalWrite(4,0);

digitalWrite(5,0);

// LEVEL

pinMode(6,INPUT);

timer=new QTimer(this);

connect(timer,SIGNAL (timeout()),this,SLOT(read\_dht11\_dat()));

timer->setSingleShot(false);

updatetimer=new QTimer(this);

connect(updatetimer,SIGNAL(timeout()),this,SLOT(update()));

dbupdatetimer=new QTimer(this);

connect(dbupdatetimer,SIGNAL(timeout()),this,SLOT(dbupdate()));

sval=new QTimer(this);

connect(sval,SIGNAL(timeout()),this,SLOT(statusval()));

QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("www.ictrlinnovations.xyz");

db.setPort(3306);

db.setDatabaseName("ictrlinn\_agribotdb");

db.setUserName("ictrlinn\_agrime");

db.setPassword("agripassword123");

if (!db.open())

{

qDebug() << "Database error occurred"<<db.lastError();

delete ui;

}

else

{

qDebug()<<"Database connected";

qDebug()<<db.databaseName();

}

timer->start(1000);

updatetimer->start(1000);

sval->start(1000);

dbupdatetimer->start(10000);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::update()

{

//qDebug() << "Update";

d1=getCM(28,29);

t=mcp3208->read\_mcp3208\_adc(0);

levelval= digitalRead(6);

if(levelval==1){

leveldata="HIGH";

}

else{

leveldata="LOW";

}

qDebug()<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

qDebug()<<"Distance:"<< d1;

qDebug()<<"Soil Moisture :"<<t;

qDebug()<<"LEVEL VALUE :"<< levelval;

qDebug()<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

}

void MainWindow::dbupdate()

{

qDebug()<<" UPDATING TO DB...";

QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("www.ictrlinnovations.xyz");

db.setPort(3306);

db.setDatabaseName("ictrlinn\_agribotdb");

db.setUserName("ictrlinn\_agrime");

db.setPassword("agripassword123");

if (!db.open())

{

qDebug() << "Database error occurred"<<db.lastError();

delete ui;

}

else

{

qDebug()<<"Database connected";

qDebug()<<db.databaseName();

QSqlQuery query;

//query.exec("insert into hrvdata(pulse) values('"+QByteArray::number(hb1)+"')");

// query.exec("UPDATE `agridata` SET `level`="+leveldata+",`ultra`='"+QByteArray::number(d1)+"',`soil`='"+QByteArray::number(t)+"',`temp`='"+QByteArray::number(dht11\_dat[0])+"',`humid`='"+QByteArray::number(dht11\_dat[2])+"' WHERE Sno=1");

query.exec("UPDATE `agridata` SET `level`='"+leveldata+"',`soil`='"+QByteArray::number(t)+"',`ultra`='"+QByteArray::number(d1)+"',`temp`='"+QByteArray::number(dht11\_dat[2])+"',`humid`='"+QByteArray::number(dht11\_dat[0])+"' WHERE Sno=1");

SleeperThread::msleep(200);

}

}

void MainWindow::statusval()

{

// qDebug() << "STATUS VALUE FUNCTION";

QSqlQuery query;

query.exec("SELECT `Sno`,`opid` FROM `agridata` WHERE Sno=1");;

if (!query.isActive())

QMessageBox::warning(this, tr("Database Error"),query.lastError().text());

while(query.next())

{

qDebug() << "Sno:" << query.value(0).toInt()

<<"Opid:"<<query.value(1).toInt();

// qDebug() << " QUERY1 VALUE" << query.value(1).toInt();

if(query.value(1).toInt()==0)

{

qDebug() << " STOP";

stop();

}

else if(query.value(1).toInt()==1)

{

qDebug() << " FRONT";

forward();

}

else if(query.value(1).toInt()==2)

{

qDebug() << " BACK";

backward();

}

else if(query.value(1).toInt()==3)

{

qDebug() << "RIGHT";

right();

}

else if(query.value(1).toInt()==4)

{

qDebug() << " LEFT";

left();

}

else if(query.value(1).toInt()==5)

{

if(soilmotor==0){

soilmotor=1;

qDebug() << " SOIL MOTOR DOWN";

soilmotodown();

}

}

else if(query.value(1).toInt()==6)

{

if(soilmotor==1){

soilmotor=0;

qDebug() << "SOIL MOTOR UP";

soilmotoup();

}

}

}

}

void MainWindow::setup\_hcsr04(int trig,int echo)

{

qDebug() << "setup";

pinMode(trig,1);//o/p

pinMode(echo,0);//i/p

//trig pin must start low

digitalWrite(trig,0);

delay(30);

}

int MainWindow::getCM(int T ,int E )

{

qDebug() << "getcm";

//Send trig pulse

digitalWrite(T, 1);

delayMicroseconds(20);

digitalWrite(T, 0);

//Wait for echo start

while(digitalRead(E) == 0);

//Wait for echo end

long startTime = micros();

while(digitalRead(E) == 1);

long travelTime = micros() - startTime;

if(travelTime>23200)

travelTime=23200;

//qDebug()<<"travel time:"<<travelTime;

//Get distance in cm

int distance = travelTime / 58;

return distance;

}

void MainWindow::read\_dht11\_dat()

{

timer->stop();

uint8\_t laststate = HIGH;

uint8\_t counter = 0;

uint8\_t j = 0, i;

float f; /\* fahrenheit \*/

dht11\_dat[0] = dht11\_dat[1] = dht11\_dat[2] = dht11\_dat[3] = dht11\_dat[4] = 0;

/\* pull pin down for 18 milliseconds \*/

pinMode( DHTPIN, OUTPUT );

digitalWrite( DHTPIN, LOW );

delay( 18 );

/\* then pull it up for 40 microseconds \*/

digitalWrite( DHTPIN, HIGH );

delayMicroseconds( 40 );

/\* prepare to read the pin \*/

pinMode( DHTPIN, INPUT );

/\* detect change and read data \*/

for ( i = 0; i < MAXTIMINGS; i++ )

{

counter = 0;

while ( digitalRead( DHTPIN ) == laststate )

{

counter++;

delayMicroseconds( 1 );

if ( counter == 255 )

{

break;

}

}

laststate = digitalRead( DHTPIN );

if ( counter == 255 )

break;

/\* ignore first 3 transitions \*/

if ( (i >= 4) && (i % 2 == 0) )

{

/\* shove each bit into the storage bytes \*/

dht11\_dat[j / 8] <<= 1;

if ( counter > 16 )

dht11\_dat[j / 8] |= 1;

j++;

}

}

/\*

\* check we read 40 bits (8bit x 5 ) + verify checksum in the last byte

\* print it out if data is good

\*/

if ( (j >= 40) &&

(dht11\_dat[4] == ( (dht11\_dat[0] + dht11\_dat[1] + dht11\_dat[2] + dht11\_dat[3]) & 0xFF) ) )

{

f = dht11\_dat[2] \* 9. / 5. + 32;

printf( "Humidity = %d.%d %% Temperature = %d.%d \*C (%.1f \*F)\n",

dht11\_dat[0], dht11\_dat[1], dht11\_dat[2], dht11\_dat[3], f );

//ui->textEdit->append(QString::number(dht11\_dat[0]));

// ui->textEdit\_2->append(QString::number(dht11\_dat[2]));

}else {

printf( "Data not good, skip\n" );

}

delay(500);

timer->start(300);

}

void MainWindow::right()

{

digitalWrite(0,0);

digitalWrite(1,1);

digitalWrite(2,1);

digitalWrite(3,0);

}

void MainWindow::left()

{

digitalWrite(0,1);

digitalWrite(1,0);

digitalWrite(2,0);

digitalWrite(3,1);

}

void MainWindow::forward()

{

digitalWrite(0,1);

digitalWrite(1,0);

digitalWrite(2,1);

digitalWrite(3,0);

}

void MainWindow::backward()

{

digitalWrite(0,0);

digitalWrite(1,1);

digitalWrite(2,0);

digitalWrite(3,1);

}

void MainWindow::stop()

{

digitalWrite(0,HIGH);

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

}

void MainWindow::soilmotodown()

{

digitalWrite(4,1);

digitalWrite(5,0);

SleeperThread::msleep(5000);

digitalWrite(4,0);

digitalWrite(5,0);

}

void MainWindow::soilmotoup()

{

digitalWrite(4,0);

digitalWrite(5,1);

SleeperThread::msleep(5000);

digitalWrite(4,0);

digitalWrite(5,0);

}