API-level Arduino Educational Board emulator

2018-09-25

Author: Xugang Wang

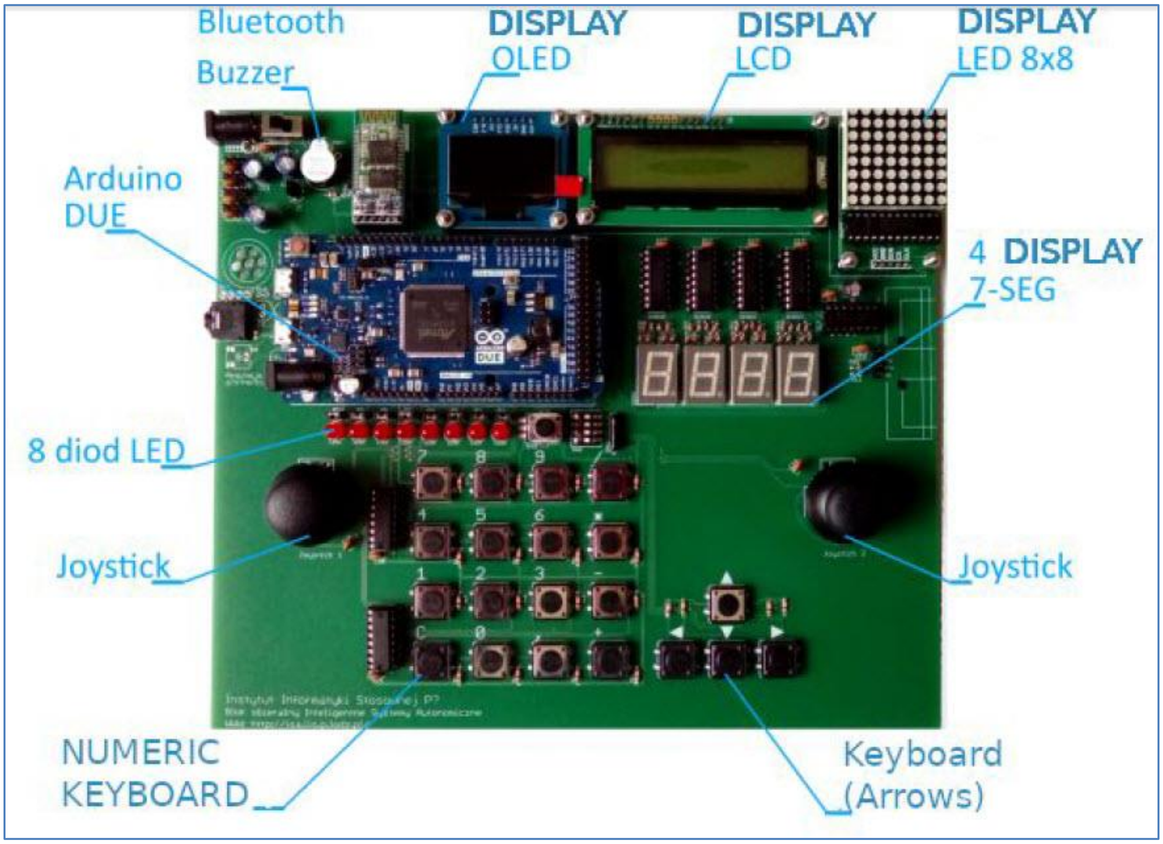
E-mail: xkingfly@hotmail.com

Part I : Introduction to the program

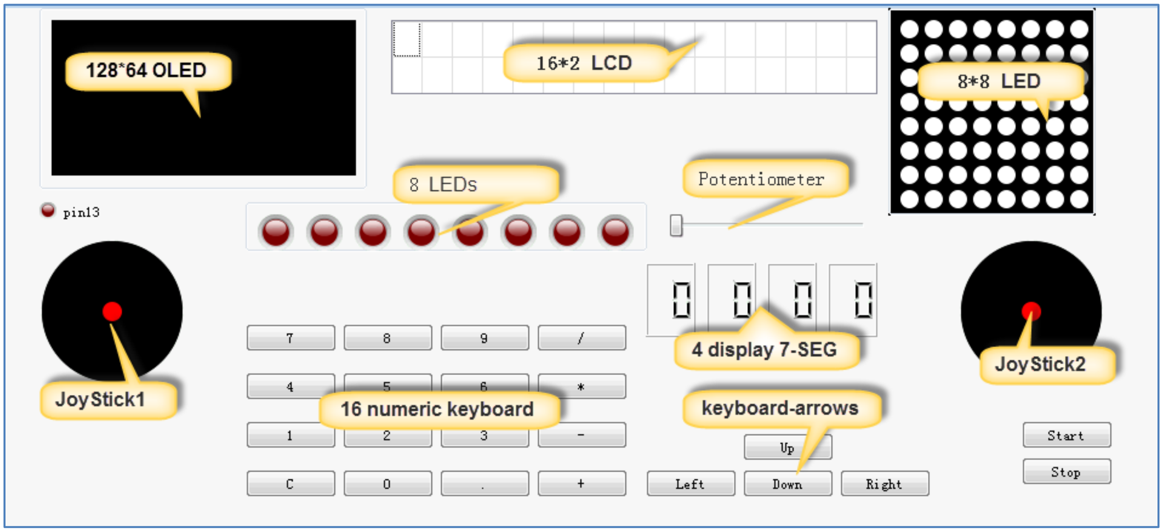
This program is built for an API-level Arduino Educational Board emulator with Qt Creator. The purpose is to allow people who will program with Arduino IDE can write the same code in this emulator and the project will perform exactly the same as Arduino Educational Board.

Before this I hope you have read the *Introduction to programming Arduino based Educational Board* written by Dr. Piotr Duch, and Dr. Tomasz Jaworski. See https://github.com/InteligentneSystemyAutonomiczneIIS/ISAFirmware

Arduino Educational Board



Qt Program emulates Arduino Educational Board



128*64 OLED:

Using OLED_X*OLED_Y (128*64 defined in isadefiniton.h) labels which are set with bitmap 0 (turn on) or 1(turn off), and set grid layout to a QGroopBox.

(Because I think 128*64 labels are too many in the UI and here has an alert" QWidget::repaint: Recursive repaint detected" , so I also write a method using paintEvent to paint bitmaps to emulate OLED only different in main window Initialization and renderAll() function. Of cause you should use only one way.)

Functions declared in ISAOLED.h

16*2 LCD:

Using a QTableWidget(rowCount 2, columnCount 40) and display only 16 columns.

Functions declared in ISALiquidCrystal.h

8*8 LED:

Using a 8*8 labels which are set with pixmap white-dot(turn off), or red-dot(turn on).

Functions declared in ISALedControl.h

8 LEDs:

Using 8 labels which are set with pixmap maroon-dot(turn off) , or pink-dot(Input model and turn on), red-dot(Output model and turn on).

Potentiometer:

Using a horizontal slider with(minimum 0, maximum 1023).

JoyStick1:

Using a red-dot-label named *JOY1* on a black-dot-label named *joyLabel1*. *JOY1* promote to " LeftJoy.h", over-write *mouseMoveEvent*, *mousePressEvent*, and *mouseReleaseEvent*.

See LeftJoy.h

JoyStick2:

Using a red-dot-label named *JOY2* on a black-dot-label named *joyLabel2*. *JOY2* promote to " RightJoy.h", over-write *mouseMoveEvent*, *mousePressEvent*, and *mouseReleaseEvent*.

See RightJoy.h

4 Display 7-SEG:

Using 4 QLCDNumber with(digitCount 2).

Functions declared in ISA7SegmentDisplay.h

16 numeric keyboard:

Using 16 labels.

Functions declared in ISAButtons.h

Keyboard-arrows:

Using 4 labels.

Below all the functions declared in library ISAxxx.h are defined in mainwindow.cpp.

ISADefinitions.h

define key-words used in this program.

ISALiquidCrystal.h

Library **ISALiquidCrystal** declare Class **ISALiquidCrystal** which is responsible for handling LCD display.

Methods available in class ISALiquidCrystal:

void begin();

void print(QString data);

void print(int data);

void print(char data);

void print(double data);

void clear();

void setCursor(int col, int row);

● *void begin()* - initialization of LCD display . It should be called in setup() function

Here set QTableWidget item(0,0) and *turnOnLCD*=true. And be notice when programing with Arduino Educational board and not called this function, the LCD will work only in the first row with black background and the function setCursor() will work only for the first parameter Col, the row will be const 0. So, in this program, this function must be called, while not, tell user to call it.(Using QMessageBox).

● *void print(QString data)* - displays the message passed in the parameter *data* on LCD display. Using QTableWidget:: setItem.

● *void print(char data)* - displays the message passed in the parameter *data* on LCD display. Using QTableWidget:: setItem.

● *void print(int data)* - convert *data* to QString and call function *print(QString data)*.

● *void print(double data)* - convert *data* to QString and call function *print(QString data)*.

● *void setCursor(int col, int row)* - sets cursor in a given position. *col* - the column at which to position the cursor, *row* - the row at which to position the cursor.

● *void clear()* - clears LCD display (Using QTableWidget::clearContents) and sets the cursor in the left upper corner (0, 0).

ISAButtons.h

The **ISAButtons** library declare Class **ISAButtons** which is responsible for buttons handling (there are 16 buttons available on numeric keypad, buttons are numbered from 0 to 15).

Methods available in class ISAButtons:

```
void init();
bool buttonPressed( int );
bool buttonReleased( int );
bool buttonState( int );
```

●*void init()* - initialization method for buttons. It should be called in a function *setup()* . Here set *turnOnButtons*=true which is declared in mainwindow.h as bool, when set true the 16 buttons work to be pressed.

●*bool buttonPressed(int)* - checks if the button with the given index has been pressed. Returns **true** if the button has been pressed otherwise returns **false**. Using *numbuttonsPreLastStates[i]* and *numbuttonsPreStates[i]* declared in mainwindow.h and value changed in slot: *numButtonPressed()*.

●*bool buttonReleased(int)* - checks if the button with the given index has been released. Returns **true** if the button has been released otherwise returns **false**. Using *numbuttonsRelLastStates [i]* and *numbuttonsRelStates [i]* declared in mainwindow.h and value changed in slot: *numButtonReleased()*.

●*bool buttonState(int)* - This method checks state of the button with the given index. Returns true if the button has been pressed otherwise returns false. Using *numbuttonsPressed[i]* declared in mainwindow.h and value changed in slots: *numButtonReleased()* and *numButtonPressed()*.

ISA7SegmentDisplay.h

Library **ISA7SegmentDisplay** declare Class **ISA7SegmentDisplay** which is responsible for handling seven-segment display.

Methods available in class ISA7SegmentDisplay:

```
void init();
void displayDigit(byte digit, int dispID, bool dot = false);
void setLed(byte values, int dispID);
```

●*void init()* - initialization of seven-segment display. The method should be called in the function *setup()*. Here set *turnOn7Seg* =true which is declared in mainwindow.h as bool.

●*void displayDigit(byte digit, int dispID, bool dot = false)* - displays the digit on the display. The type *byte* is defined in ISADefinitions.h as *typedef uint8_t byte*. *digit* - the digit to be displayed, must be in the range <0, 9>, in other case function does nothing, *dispID* -

identification number of display, on which the number should be displayed, must be a value in the range <0,4), in other case functions does nothing, *dot* - pass information whether to display a dot or not, by default is set to false. Here using the QLCDNumber:: display.

●***void setLed(byte values, int dispID)* – Sorry, here do nothing**. In the Arduino board it will set state of LED on a given display. But QLCDNumber do not support this, maybe I can use a third-part widget later.

ISALedControl.h

Library **ISALedControl** declare Class **ISALedControl** which is responsible for handling 8x8 LED matrix.

Methods available in class ISALedControl:

void init();

void clearDisplay( );

void setRow(int row, byte value);

void setColumn(int col, byte value);

void setLed(int row, int col, bool value);

●*void init()* - initializes 8x8 LED matrix. Method should be called in *setup* () function. Here set *turnOnLedDisplay*=true which is declared in mainwindow.h as bool.

●*void clearDisplay( )* - turns off all diodes on the display. Here use *setPixmap* function to set the labels with a white-dot pixmap.

●*void setRow(int row, byte value)* - changes the state of diodes in one row. *row* - row id, *value* - eight bit value, in which each bit set on 1 turns on corresponding diode, and each bit set on 0 turns off diode. Here set label with a red-dot pixmap while bit is 1, and set label with a white-dot pixmap while bit is 0.

●*void setColumn(int col, byte value)* - changes the state of diodes in the given column. *col* - column id, *value* - eight bit value, in which each bit set on 1 turns on corresponding diode, and each bit set on 0 turns off diode. Here set label with a red-dot pixmap while bit is 1, and set label with a white-dot pixmap while bit is 0.

●*void setLed(int row, int col, bool value)* - changes the state of the diode. *row* - row id (from range <0; 7>), *col* - column id (from range <0; 7>), *state* - state of the diode (**true** -

turned on - set label with a red-dot pixmap, **false** - turned off- set label with a white-dot pixmap).

ISAOLED.h

Library **ISAOLED** declare Class **ISAOLED** which is responsible for handling OLED. ***Sorry, because the time is limited, I have not finished all of the functions emulate as the same behavior as on Arduino Educational Board, such as when you call print(data) function after writeRect(x, y, w, h, fill) function it has two different behaviors under the rectangular at position(x, 8*(y/8+1)), when(y+ h) is less than 8*(y/8+1), otherwise follow it at the end of the rectangular at (y + h)/8.***

In this vision programing with OLED you should use clear() function in loop(), and use gotoXY() function to set the right position you want to print text using print() function.

Here use a *bitmap* array (of size 128 columns and 64 rows) to execution of simple graphical operations. After performing graphical operations one should execute method *renderAll(),* which will read the bitmap array and then set values to displayed on the OLED.

Methods available in class ISAOLED:

void begin();

void clear(bool render = true);

void write(byte data);

void gotoXY(int cx,int cy);

void setPixel(int x, int y, bool v);

void writeLine(int x1,int y1,int x2,int y2);

void writeRect(int x, int y, int w, int h, bool fill=false);

void print(QString text);

void print(int x);

void print(double x);

void renderAll();

●*void begin()* - initializes OLED display of resolution 128x64. Method should be called in *setup ()* function. Here set *oledCol*=0 and *oledRow*=0 which are declared in mainwindow.cpp as int.

●*void clear(bool render = true)* - turns off all pixels on the display. *render* - logical value resulting in immediate removal of image from display. Here set the *bitmap* array values as *bitmapBB* which is a bitmap values 1.

●*void write(byte data)* - displays a character based on ASCII code. *data* - ASCII code of a symbol, that is going to be displayed, non-ASCII symbols will not be displayed. Here using *static const uint8_t ASCII[][5]* declared in ISADefinitions.h to compare each bit with *data* and then set bitmap values as *bitmapWW* which is a bitmap values 0.

●*void gotoXY(int cx, int cy)* - sets a cursor for displaying text in position *c x , c y* . *cx* - coordinate on X axis, on which the cursor is set (accepts values from range <0; 127>), *cy* - coordinate on Y axis, on which the cursor is set (accepts values from range<0; 7>). Here set *oledRow=cy*8* and *oledCol=cx*.

●*void setPixel(int x, int y, bool v)* - changes state of the given pixel. *x , y* - coordinates of the pixel, *v* - state of the pixel (**true** - turned on, set *bitmap[y][x]=bitmapWW,* ***false***- turned off, set *bitmap[y][x]=bitmapBB*).

●*void writeLine(int x1,int y1,int x2,int y2)* - draws only vertical or horizontal lines. *x1, y1* - coordinates of beginning of the segment, *x2, y2* - coordinates of end of the segment. Here call *writeRect()* function.

●*void writeRect(int x, int y, int w, int h, bool fill=false)* - draws a rectangle. *x, y* - coordinates of upper left corner of the rectangle, *w* - width of the rectangle, *h* - height of the rectangle, *fill* - flag indicates whether the rectangle will be filled or not(**true** - filled, **false** - only borders). Also using set *bitmap* values as *bitmapWW.*

●*void print(QString text)* - displays text on the display. *text* - text, that is going to be displayed. Here call *write()* function.

●*void print(int x)* - displays text on the display. *x* - digit, that is going to be displayed. Here convert *x* to QString and call function *print(QString text).*

●*void print(double x)* - displays text on the display. *x* - digit, that is going to be displayed. Here convert *x* to QString and call function *print(QString text).*

●*void renderAll()*- sends built frame of image to the display and displays it immediately. Here set the 128*64 labels *oledDot* array each item using *setPixmap()* function with *bitmap* array. As mentioned before, another way is to use *paintEvent()* function paint *bitmap* array each item bitmap at the give positon.

LeftJoy.h

Library **LeftJoy** declare Class **LeftJoy** which inherit from QLabel is responsible for handling emulating joystick. Here rewrite *mouseMoveEvent*, *mousePressEvent, mouseReleaseEvent.*

*void mousePressEvent(QMouseEvent*e);*

*void mouseMoveEvent(QMouseEvent*e);*

*void mouseReleaseEvent(QMouseEvent *e);*

●*void mousePressEvent(QMouseEvent*e)* – initializes circle which *JOY1* will move within. Circle center as the *JOY1* center, and radius as *joyLabel1*'s width/2-10.

●*void mouseMoveEvent(QMouseEvent*e)* – control the *JOY1* move within the circle(max-the *JOY1*'s center at the circle's arc). And set the *joy1rx* and *joy1ry* value which can be read using *analogRead()* function.

●*void mouseReleaseEvent(QMouseEvent *e)* – set *JOY1* move to the center of *joyLabel1*.

RightJoy.h

Library **RightJoy** declare Class **RightJoy** which inherit from QLabel is responsible for handling emulating joystick. Here rewrite *mouseMoveEvent, mousePressEvent, mouseReleaseEvent.*

*void mousePressEvent(QMouseEvent*e);*

*void mouseMoveEvent(QMouseEvent*e);*

*void mouseReleaseEvent(QMouseEvent *e);*

●*void mousePressEvent(QMouseEvent*e)* – initializes circle which *JOY2* will move within. Circle center as the *JOY2* center, and radius as *joyLabel2*'s width/2-10.

●*void mouseMoveEvent(QMouseEvent*e)* – control the *JOY2* move within the circle(max-the *JOY2*'s center at the circle's arc). And set the *joy2rx* and *joy2ry* value which can be read using *analogRead()* function.

●*void mouseReleaseEvent(QMouseEvent *e)* – set *JOY2* move to the center of *joyLabel2*.

MainWindow.h

Library **MainWindow** declare Class **MainWindow** which is default for the Qt GUI Application. The main methods declared here are:

Public slots:

●*void delay(int ms)*;//using a timer to emulate delay function in Arduino. *ms* milliseconds. **Connect to UserCode's signal *void delay(int ms)*.**

●*void pinMode(int i, int mode);//*using some bool arrays to emulate pinMode function in Arduino. *i*-pin number; mode *INPUT, OUTPUT.* Here control the 8LEDs lights and 4Arrow-keys. **Connect to UserCode's signal *void pinMode(int i, int mode)*.**

●*void digitalWrite(int i, bool value);//*emulate the digitalWrite function in Arduino. *i* –pin number; value *HIGH, LOW.* Here control the 8LEDs lights and pin13 light. Using QLabel setPixmap function. **Connect to UserCode's signal *void digitalWrite(int i, bool value)*.**

●*bool digitalRead(int i);//i* pin-number, return *HIHG, LOW.* Emulate digitalRead function in Arduino. Here only read from 4Arrow-keys. **Connect to UserCode's signal *bool digitalRead(int i)*.**

●*int analogRead(int i);//i-* pin number, return 0-1023. Here only work with POT, JOY1X, JOY1Y, JOY2X, JOY2Y all defined in **ISADefinitions.h**. Emulate analogRead function in Arduino. **Connect to UserCode's signal *int analogRead(int i)*.**

●*void analogWrite(int i, int value);//i* –pin number, value from 0 to 255. Emulate analogWrite function in Arduino. Here only control 8LEDs, depended on the *value* and the call pinMode and digitalWrite function. Such as when value is 255, call *pinMode(i, OUTPUT)* and then call *digitalWrite(i, HIGH)*. **Connect to UserCode's signal *void analogWrite(int i, int value)*.**

And 6 other slots using for 4Arrow-keys or 16Number-buttons

*void numButtonPressed();//*16 number buttons on press slot

void numButtonReleased();//16 number buttons on release slot

*void numButtonClicked();//*16 number buttons on click slot

*void keyArrowButPressed();//*4 key arrow buttons on press slot

*void keyArrowButReleased();//*4 key arrow buttons on release slot

*void keyArrowButClicked();//*4 key arrow buttons on click slot

And 4 public functions:

*void quitExe();//*used in stop-close button to quit application

*void keyPressEvent(QKeyEvent *key);* //key_W, key_A, key_S, key_D, key_UP, key_DOWN, key_LEFT, key_RIGHT

*void keyReleaseEvent(QKeyEvent *key);* //key_W, key_A, key_S, key_D, key_UP, key_DOWN, key_LEFT, key_RIGHT

*void paintEvent(QPaintEvent *e);*//as mentioned this is another way to emulate OLED. Take care to use it.

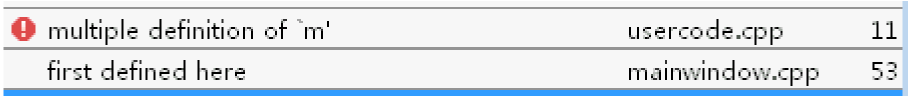
Part II : How to emulate Arduino programing with it

In the project there are two files allow user to write their own codes to emulate Arduino programing, usecode.h and usecode.cpp.

In the Arduino project, there are two basic functions, setup() and loop(). And in this project we have UserCode::setup() and UserCode::loop(). Just write your code in UserCode::setup() and UserCode::loop() as the same you write in Arduino setup and loop functions. Other functions you define in Arduino project, you should declare in usercode.h and then define in usercode.cpp

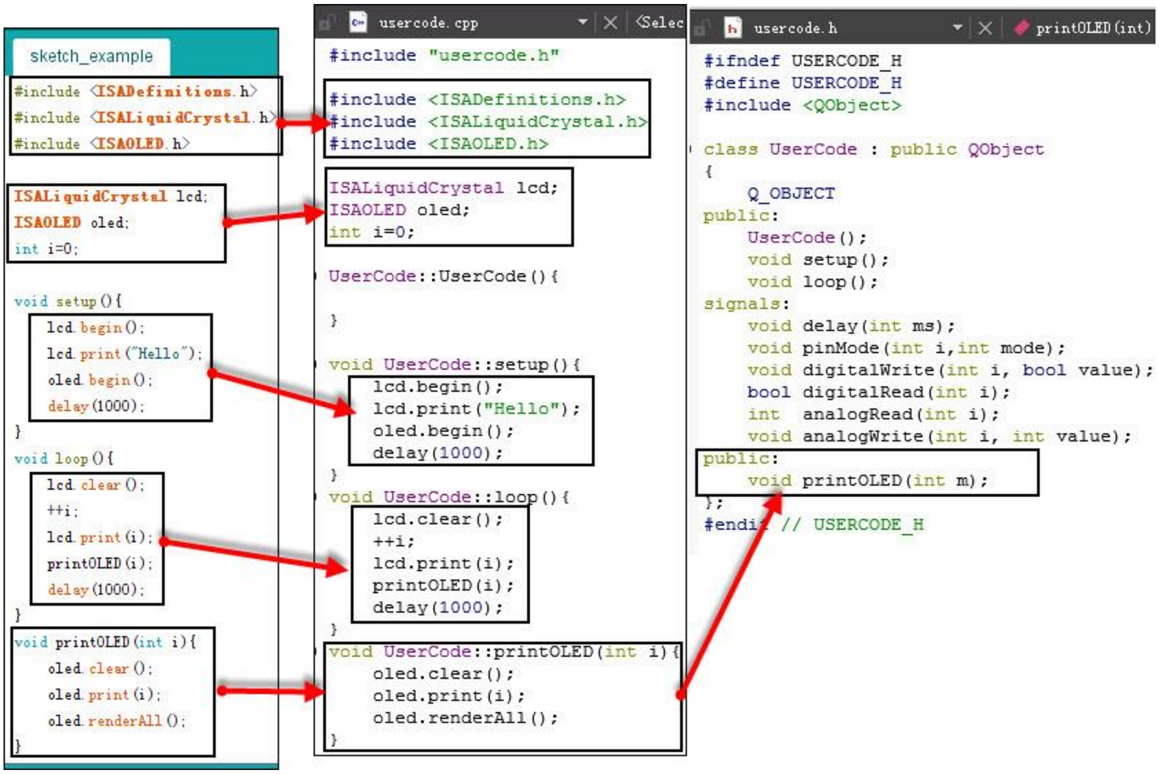
Please be advised that you can only add your own code under UserCode's public function block. Do Not delate or modify any other code.

Some variables have been defined in this project(such as "m" defined as MainWindow, "r" defined as radius used in Joystick functions), so please use a different variable in your code. If not, program will give the error alerts and you should change your code. Such as:



```
multiple definition of `m'      usercode.cpp      11
first defined here              mainwindow.cpp    53
```

Blew is an example how to rewrite your code from Arduino IDE to this program and vice versa.



```
#include <ISADefinitions.h>
#include <ISALiquidCrystal.h>
#include <ISAOLED.h>

ISALiquidCrystal lcd;
ISAOLED oled;
int i=0;

void setup(){
    lcd.begin();
    lcd.print("Hello");
    oled.begin();
    delay(1000);
}
void loop(){
    lcd.clear();
    ++i;
    lcd.print(i);
    printOLED(i);
    delay(1000);
}
void printOLED(int i){
    oled.clear();
    oled.print(i);
    oled.renderAll();
}
```

```
#include "usercode.h"

#include <ISADefinitions.h>
#include <ISALiquidCrystal.h>
#include <ISAOLED.h>

ISALiquidCrystal lcd;
ISAOLED oled;
int i=0;

UserCode::UserCode(){

}

void UserCode::setup(){
    lcd.begin();
    lcd.print("Hello");
    oled.begin();
    delay(1000);
}
void UserCode::loop(){
    lcd.clear();
    ++i;
    lcd.print(i);
    printOLED(i);
    delay(1000);
}
void UserCode::printOLED(int i){
    oled.clear();
    oled.print(i);
    oled.renderAll();
}
```

```
#ifndef USERCODE_H
#define USERCODE_H
#include <QObject>

class UserCode : public QObject
{
    Q_OBJECT
public:
    UserCode();
    void setup();
    void loop();
signals:
    void delay(int ms);
    void pinMode(int i,int mode);
    void digitalWrite(int i, bool value);
    bool digitalRead(int i);
    int  analogRead(int i);
    void analogWrite(int i, int value);
public:
    void printOLED(int m);
};
#endif // USERCODE_H
```