Internet of Things- Arduino Final Project

Layla de Souza Barbosa

Lizandra Esterque

Georgian At Ilac - Computer Programming Diploma

I. Project Overview

The Wake to Sunrise Alarm Clock is a Arduino program simulating a digital clock with a alarm function attached where a RGB light simulates the sunlight, making it easier to human body to wake. This project aims to fulfill requirements for the final project in Internet of Things using Arduino course for Computer Programming Diploma.

The development environment used to create the circuit was the Tinkercad website and the code was written using C++ language, To create a functional project we made use of a series of components, as listed:

- 2 Arduino UNO R3 connected trough the master-slave system;
- 2 bread boards used to organize the wires and components;
- A Liquid Crystal Display 16x2;
- A piezo buzzer;
- A RGB LED;
- 2 pushbuttons;
- An IR remote and sensor;
- Resistors and wires as needed to connect components to both Arduino UNO.

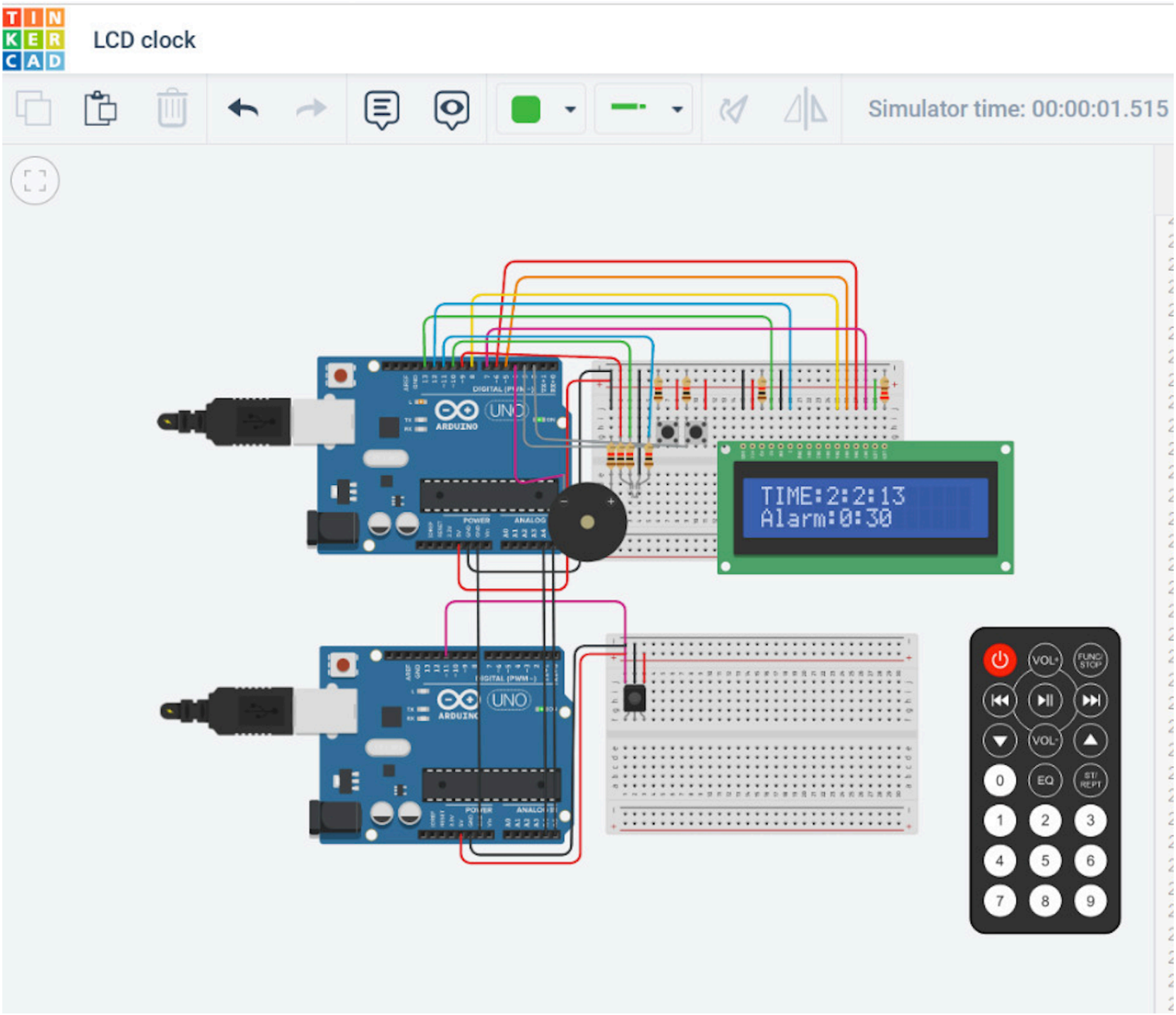
In addition, we imported some libraries that allowed the use of methods to work with the components selected, such as:

• Wire.h : allows you to communicate with I2C/TWI devices, opening the communication door between 2 Arduinos UNO connected between themselves.

• IRremote.h: send and receive infrared signals with multiple protocols, allowing us to use the remote to control the alarm settings

• LiquidCrystal.h: allows an Arduino board to control LiquidCrystal displays, making possible to actually show the user the time and alarm that is set up.

II. Circuit set up



III. Code

A. Clock_1:

```
/*C++
Authors: Lizandra Esterque & Layla Barbosa
Date: 12/10/2022
Slave clock: This board has the lcd to show time and alarm time set up, the rib light that turns
on with a buzzer that plays a warm song when the time and alarm time are the same. To control
the 2 variables, hours and minutes, it has 2 push buttons, one for each variable.
*/

// importing libraries
#include <LiquidCrystal.h>
#include <Wire.h>

//public constants
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
```

```
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

```
#define REST      0

//instatianting variables and an array for the alarm song
int tempo = 180;
int melody[] = {
  NOTE_E5, 8, NOTE_D5, 8, NOTE_FS4, 4, NOTE_GS4, 4,
  NOTE_CS5, 8, NOTE_B4, 8, NOTE_D4, 4, NOTE_E4, 4,
  NOTE_B4, 8, NOTE_A4, 8, NOTE_CS4, 4, NOTE_E4, 4,
  NOTE_A4, 2,
};
int notes = sizeof(melody) / sizeof(melody[0]) / 2;
int wholenote = (60000 * 4) / tempo;
int divider = 0, noteDuration = 0;

//instatianting variables for lcd, rgbled and pushbuttons
LiquidCrystal lcd (13,12,8,5,6,7);
int RedPin = 9;
int GreenPin =10;
int BluePin = 11;
int buzzerPin = 4;
int buttonpin = 3;
int buttonpin2=2;

//instatianting variables for seconds, hour and minutes for the clock;
int h =1;
int m =0;
int s;
//and minutes and hours for alarm
int hA;
int mA;


void setup()
{
        //inititalizing all components and communication doors
        Wire.begin(1);
        Wire.onReceive(receiveEvent);
        Serial.begin(9600);
        lcd.begin (16,2);
        pinMode(RedPin, OUTPUT);
        pinMode(GreenPin, OUTPUT);
        pinMode(BluePin, OUTPUT);
        pinMode(buzzerPin, OUTPUT);
        //defining function for the pushbuttons
        attachInterrupt(digitalPinToInterrupt(buttonpin), hours, RISING);
        attachInterrupt(digitalPinToInterrupt(buttonpin2), minu, RISING);
}

//function that receives information from the master
void receiveEvent(int bytes) {
mA = Wire.read();

}
```

```
//functions to define hour and minute
void hours () {
          h++;
}

void minu (){
          m++;
}


void loop(){
          //starting the rub turned off
          analogWrite(RedPin,0);
          analogWrite(GreenPin,0);
          analogWrite(BluePin,0);
          lcd.clear();

          //attributing the value received from the master to a inside variable
          int Time = mA;

          //if controllers for set up the alarm in 60 minutes intervals and 24 hours
          if (Time>=60) {
                    Time=0;
                    hA++;
                    if (hA==24) {
                              hA=0;
                    }
          }

          if (Time<0) {
                    Time=45;
                    hA--;
                    if (hA==-1){
                              hA=23;
                     }
          }

          //printing the value on the monitor serial for better following the program flow
          Serial.print(mA);
          //printing the timed alarm time on the lcd
          lcd.setCursor(0, 0);
          s++;
          lcd.print("TIME:" );
          lcd.print(h);
          lcd.print(":");
          lcd.print(m);
          lcd.print(":");
          lcd.print(s);

          lcd.setCursor(0, 1);
          lcd.print("Alarm:" );
          lcd.print(hA);
          lcd.print(":");
          lcd.print(Time);
```

```
        delay(100);
        //get the time rolling as a clock
        if (s == 60){
                s = 0;
                m++;
        }
        if (m == 60){
                m = 0;
                 h = h++;
         }
        if (h == 24){
                h = 0;
         }

        //checking if the alarm should goes off
         if (h == hA && m == Time) {
                Alarm();}
        }
 //function for when the alarm goes off
void Alarm () {
        //turn on the rgbled
        analogWrite(RedPin,255);
        analogWrite(GreenPin,255);
        analogWrite(BluePin,255);
        //play the song
        for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
                 // if controller to define melody
                divider = melody[thisNote + 1];
                if (divider > 0) {
                         //regular note, just proceed
                        noteDuration = (wholenote) / divider;
                }
                else if (divider < 0) {
                        // dotted notes are represented with negative durations!!
                        noteDuration = (wholenote) / abs(divider);
                        noteDuration *= 1.5; // increases the duration in half for dotted notes
                 }
                //playing each note in the buzzer
                tone(buzzerPin, melody[thisNote], noteDuration * 0.9);
                delay(noteDuration);
                noTone(buzzerPin);
        }
 }
```

B. Clock_2:

```
/*C++
Authors: Lizandra Esterque & Layla Barbosa
Date: 12/10/2022
Master clock: This board has the IR remote that controls the alarm with increments of 15
minutes, also sends the message for the slave to set up the alarm
*/

// importing libraries
#include <Wire.h>
#include <IRremote.h>

//instatianting variables for the remote
int RECV_PIN = 11; // the pin where you connect the output pin of IR sensor
IRrecv irrecv(RECV_PIN);
decode_results results;
//ant variables to store information further sent to the slave
int value = 0;
int mA;


void setup(){
	//opening communications doors and initializing components
	Wire.begin();
	Serial.begin(9600);
	irrecv.enableIRIn();
}

void loop(){
	//transmitting information to slave
	Wire.beginTransmission(1);
	Wire.write(mA);
	Wire.endTransmission();

	//reading input from remote
	 if (irrecv.decode(&results)) {
		value = results.value;
		Serial.println(" ");
		Serial.print("Code: ");
		Serial.println(value); //prints the value a a button press
		Serial.println(" ");
		irrecv.resume(); // Receive the next value
		Serial.println("******************");
	}


	//nested if controllers for add 15 min increments to the variable in a clock wise form
	if (value == -32641) {
		mA= 15 + mA;

		 if (mA>=61) {
			mA=0;
```

```
                }
        value=0;
         }

        //nested if controllers for subtract 15 min increments to the variable in a clock wise form
         if (value == -32641) {
                mA= mA - 15;

                if (mA>=-1) {
                        mA=45;
                }
        value=0;
        }
}
```