МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики  
Кафедра суперкомпьютеров и общей информатики

**Отчет по лабораторной работе №3**

Дисциплина: «Технологии Интернета вещей»

Вариант 10

Выполнил: Мелешенко И.С.

Группа: 6133-010402D.

Дата: 16.05.2022

Самара 2022

В ходе выполнения лабораторной работы была создана модель счетчика электроэнергии. Счетчик был создан на основе микроконтроллера Arduino Uno в виртуальной среде Tinkercad.

На первом этапе лабораторной работы была собрана схема счетчика электроэнергии. Схема приведена на рисунке 1.

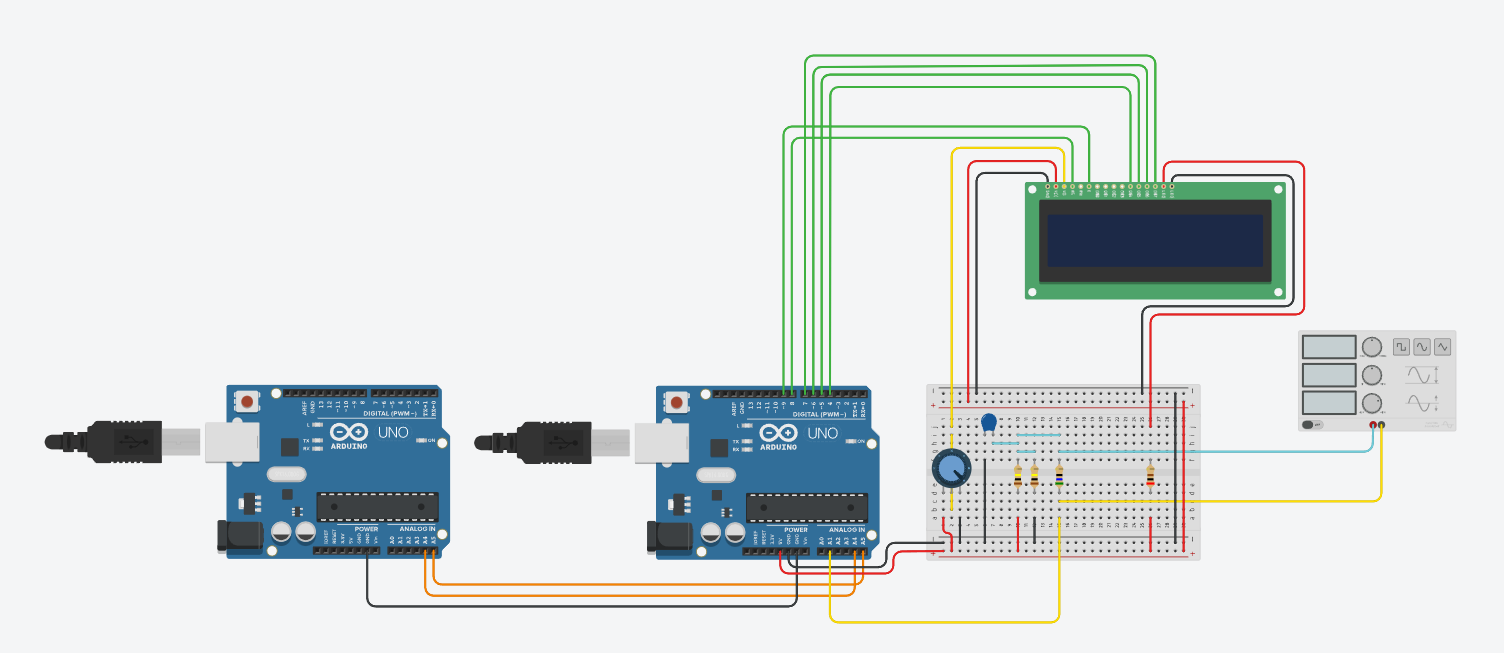


Рисунок 1 – Схема счетчика

На втором этапе были запрограммированы микроконтроллеры. Код приведен в приложении А.

В каждом скетче присутствуют две обязательные для Arduino функции *setup()* и *loop()*. Первая вызывается однократно при старте, вторая выполняется в бесконечном цикле.

Используемые в коде библиотеки:

Библиотека Wire – нужна для реализации интерфейса в среде Arduino IDE. Содержит функции: *begin()* – начало работы интерфейса; *write()* – передача данных по шине I2C; *read()* – Получаем значения по шине I2C; *endTransmission()* – завершает передачу данных ведомому устройству, которая была запущена функцией *beginTransmission()*.

Библиотека *LiquidCrystal* – нужна для работы с текстовыми жидкокристаллическими экранами. Содержит функнции: инициализации контактов *LiquidCrystal lcd(8, 9, 4, 5, 6, 7)* экрана; *begin(m,n)* (начало работы, размерность экрана по строкам и столбцам); *clear* (очистка экрана); *setCursor(x,y)* – установить курсор; *print* – вывести на дисплей.

Функция *analogRead(currentPin)* – считывает значение с указанного аналогового вывода. Платы Arduino содержат многоканальный 10-битный аналого-цифровой преобразователь. Это означает, что он будет отображать входные напряжения между 0 и рабочим напряжением (5 В или 3,3 В) в целочисленные значения от 0 до 1023. На Arduino UNO, например, это дает разрешение между показаниями: 5 вольт / 1024 единиц или , 0,0049 В (4,9 мВ) на единицу.

*IRremote* – библиотека для работы с инфракрасными датчиками, поддерживает стандартные форматы NEC, SIRC, RC5, RC6, а также может принимать "сырые" данные.

*EEPROM* (англ. Electrically Erasable Programmable Read-Only Memory – электрически стираемое перепрограммируемое постоянное запоминающее устройство (ЭСППЗУ)), она же энергонезависимая память. Простыми словами: *EEPROM* – память, к которой мы имеем полный доступ из выполняющейся программы, т.е. можем во время выполнения читать и писать туда данные, и эти данные не сбрасываются при перезагрузке МК.

Так как вычисляемая мощность является вещественным числом, а передавать мы можем только целые числа, то для передачи рассчитанных киловатт было выполнено умножение на 1000 до начала передачи и деление на 1000 после окончания передачи. Тем самым через канал передачи данных транслировались только целые числа, но при этом были сохранены знаки после запятой у рассчитанных киловатт.

Ссылка на проект: https://www.tinkercad.com/things/d1IpuUiIYOh-neat-densor-crift/editel?sharecode=Eoja04h\_QtyKR2LVm5JikzTfDur-fjeNX59gk1-urds

**Приложение А**

Код для ведущей платы:

#include <Wire.h>

#include <LiquidCrystal.h>

#include <IRremote.h>

#include <EEPROM.h>

int x;

int currentPin = 1; //Assign CT input to pin 1

double kilos = 0;

int peakPower = 0;

unsigned long startMillis;

unsigned long endMillis;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Assign LCD screen pins, as per LCD shield requirements

void setup(){

lcd.begin(16,2); // columns, rows. use 16,2 for a 16x2 LCD, etc.

lcd.clear();

lcd.setCursor(0,0); // set cursor to column 0, row 0 (the first row)

lcd.print("Arduino");

lcd.setCursor(0,1); // set cursor to column 0, row 1 (the second row)

lcd.print("Energy Meter");

startMillis = millis();

Wire.begin();

Serial.begin(5000); // Скорость передачи данных

}

void loop()

{

int current = 0;

int maxCurrent = 0;

int minCurrent = 1000;

for (int i=0 ; i<500; i++) //Monitors and logs the current input for 500 cycles to determine max and min current

{

current = analogRead(currentPin); //Reads current input and records maximum and minimum current

if(current >= maxCurrent)

maxCurrent = current;

else if(current <= minCurrent)

minCurrent = current;

}

if (maxCurrent <= 517)

{

maxCurrent = 516;

}

double RMSCurrent = ((maxCurrent - 516)\*0.707)/11.8337; //Calculates RMS current based on maximum value

int RMSPower = 230\*RMSCurrent; //Calculates RMS Power Assuming Voltage 110VAC, change to 220VAC accordingly

if (RMSPower > peakPower)

{

peakPower = RMSPower;

}

endMillis = millis();

unsigned long time = endMillis - startMillis;

kilos = kilos + ((double)RMSPower \* ((double)time/60/60/1000000)); //Calculate kilowatt hours used

startMillis = millis();

delay (1090);

lcd.clear();

lcd.setCursor(0,0); // Displays all current data

lcd.print(RMSCurrent);

lcd.print("A");

lcd.setCursor(10,0);

lcd.print(RMSPower);

lcd.print("W");

lcd.setCursor(0,1);

lcd.print(kilos);

lcd.print("kWh");

lcd.setCursor(10,1);

lcd.print(peakPower);

lcd.print("W");

x = int(kilos\*1000);

Wire.beginTransmission(10);

delayMicroseconds(1090);

Wire.write(x);

Wire.endTransmission();

Serial.println(kilos);

}

Код для ведомой платы:

// C++ code

//

#include <Wire.h>

int x = 0;

double kilos = 0;

void setup()

{

Wire.begin(10); // 10 здесь адрес Slave (упоминается также в коде основной платы)

Wire.onReceive(receiveEvent);

Serial.begin(5000); // Скорость передачи данных

}

void receiveEvent(int bytes) {

delayMicroseconds(1090); //задержка приема данных

x = Wire.read(); // Получаем значения х от основной платы

//Перевод в правильные единицы

kilos = x;

kilos = kilos/1000;

Serial.println(kilos);

}

void loop()

{

}