МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки / специальности)

Электроника и робототехника
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Электронный счетчик монет

Обучающийся С.А. Катин _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, А. В. Прядилов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант О.А. Головач
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Название дипломной работы: «Электронный счетчик монет».

Выпускная работа состоит из введения, трех частей, заключения, 35 рисунков, списка литературы, включающего 5 зарубежных источников и графической части на 6 листах формата А1.

Ключевым вопросом дипломной работы является разработка устройства электронного счетчика монет.

Объектом исследования является устройство, сочетающее в себе функции электронного счетчика монет и часов.

Задачами работы являются:

1. Обзор состояния вопроса. В рамках этого задания я проанализирую, какие аналогичные продукты и их технические решения представлены на рынке.

2. Разработка электрической схемы. В данной задаче я разработаю схему электрическую принципиальную, на которой будет показано как устройства подключаются друг к другу.

3. Написание программы для микроконтроллера. В данной задаче разберем код программы устройства.

4. Практическая реализация проекта. В последнем разделе напечатаем корпус устройства и соберем саму схему в нем.

Степень реализации проекта: теоретически рассчитаны параметры системы, разработаны схемы и подобраны элементы, написана программа для микроконтроллера, напечатан корпус устройства для 3Д принтера и собрано все устройство в одном корпусе.

Область применения данной работы – использование в повседневной жизни.

Актуальность разработки устройства электронного счетчика монет с часами заключается в отсутствие подобных устройств на рынке.

Работа представляет интерес для широкого круга людей.

Abstract

The title of the graduation work is «Electronic Coin Counter».

The senior paper consists of an introduction, three parts, a conclusion, 35 drawings, a list of references, including 5 foreign sources and a graphic part on 6 sheets of A1 sheets.

The key issue of the thesis is the design of lighting of an electronic coin counter device.

The object of research is device that combines the functions of an electronic coin counter and a watch.

The tasks of the work are:

1. A review of the status of the issue. In this task, I will analyze which similar products and their technical solutions are on the market.

2. Development of an electrical circuit. In this task, I will develop an electrical schematic diagram, which will show how the devices are connected to each other.

3. Writing a program for the microcontroller. In this task, we will analyze the program code of the device.

4. Practical implementation of the project. In the last paragraph, we will print the device case and assemble the circuit itself in it.

The degree of implementation of the project: the system parameters are theoretically calculated, schematics are developed and elements were selected, a program for the microcontroller is written, the housing of the device for a 3D printer is printed and the entire device is assembled in one housing.

The scope of this work is use in everyday life.

The relevance of developing an electronic coin counter device with a watch lies in the absence of such devices on the market.

The work is of interest to a wide range of people.

Содержание

Введение 5

1 Состояние вопроса 6

1.1 Формулирование цели и задач проекта 6

1.2 Анализ исходных данных и существующих решений 6

1.2.1 Исходные данные 6

1.2.2 Анализ существующих решений 7

2 Разработка электрической принципиальной схемы 9

2.1 Разработка системы управления 9

2.2 Написание программы для микроконтроллера 15

3 Практическая реализация проекта 26

3.1 Сборка модели 26

3.1.1 Моделирование корпуса копилки 26

3.1.2 Моделирование крышки корпуса 32

3.1.3 Моделирование монетоприемника 34

3.1.4 Сборка устройства. 36

3.2 Отладка модели 38

3.3 Экспериментальные исследования 40

3.4 Описание лабораторного стенда 40

Заключение 42

Список используемых источников 43

Приложение А Код программы 45

Приложение Б Действующий макет электронного счетчика монет. 52

Введение

В современном мир технологии развиваются очень стремительно. Также сейчас в мире все больше и больше появляется офисной работы и потому всё больше внимания уделяется созданию удобных и функциональных устройств для повседневного использования не только дома, но и на работе. Одним из таких устройств вполне может стать электронная настольная копилка с часами на базе платформы Arduino.

Целью данной выпускной квалификационной работы является разработка электронной настольной копилки с часами на базе Arduino, которая бы сочетала в себе функции копилки и часов, была проста в использовании и настройке, а также имела привлекательный дизайн.

Для достижения поставленной цели необходимо решить ряд задач, таких как:

- Анализ существующих решений
- Выбор компонентов
- Разработка системы управления
- Написание программного обеспечения для управления копилкой и часами
- Создание корпуса устройства
- Проведение испытаний готового устройства.

В ходе выполнения работы будут проанализированы существующие решения и выбраны оптимальные компоненты для реализации проекта. Особое внимание будет уделено разработке программы, которая обеспечит корректную работу копилки и часов, а также возможность их легкой настройки.

Результатом данной работы станет готовое устройство — электронная копилка с часами на базе Arduino, которое может быть использовано людьми как дома, так и в офисе.

1 Состояние вопроса

1.1 Формулирование цели и задач проекта

Тема выпускной квалификационной работы: “Электронный счетчик монет”.

Задачи ВКР:

1. Изучить теоретические основы работы с платформой Arduino и сопутствующими электронными компонентами.

2. Разработать схему подключения компонентов для создания электронной настольной копилки с часами.

3. Написать программное обеспечение для управления функциями копилки и часов.

4. Протестировать разработанное устройство на предмет корректной работы всех его функций.

1.2 Анализ исходных данных и существующих решений

1.2.1 Исходные данные

Исходными для ВКР являются следующие данные:

- Распознавание монет номиналами: 1 руб., 2 руб., 5 руб., 10 руб.

- Отображение счетчика суммы: на ЖК дисплее

- Наличие автономного питания: аккумуляторная батарея формата 18650

Исходя из этих данных нам предстоит разработать устройство, которое смогло бы распознавать все номиналы ходовых российских монет. Сумма этих монет должна выводиться на ЖК дисплей. Также на ЖК дисплее в моменты когда в копилку не кладут монеты будет отображаться дата и текущее время. Это сделает ее более практичной и полезной вещью на любом столе. По условиям питания она должна быть автономной с аккумуляторной батареей формата 18650. Однако если мы рассматриваем устройство для повседневного

применения, то также необходимо реализовать возможность питания и от сети. Совмещение обоих технических решений позволит работать устройству неограниченно долго (при питании от сети), а также переносить его с места на место без потери данных и удобно демонстрировать изделие на днях открытых дверей (при питании от аккумуляторной батареи).

1.2.2 Анализ существующих решений

На данный момент на рынке существует много различных монетоприемников. Их все можно разделить на 2 простые категории: программируемые и эталонные.

- Эталонные монетоприемники (Рисунок 1) называются так потому, что работают по принципу сравнения полученной монеты с заранее установленным образцом - эталоном. Такие монетоприемники различают только один номинал монеты или жетона. Для определения подлинности монеты используются индуктивные датчики, которые сравнивают электромагнитные параметры пролетевшего объекта с параметрами эталона. Современные монетоприемники данного типа оснащаются возвратным механизмом и защитным устройством, которые служат для предотвращения вытягивание монеты после ее пролета через монетоприемник.

Рисунок 1 – Эталонный монетоприемник

- Программируемый монетоприемник (Рисунок 2) в отличие от своего эталонного брата, способен принимать уже более одного номинала монет. Это получается за счет того, что в отличие от эталонного монетоприемника, характеристики каждой монеты, которую может распознать программируемый монетоприемник хранятся в его флеш-памяти. Данные характеристики программируются в память при установке монетоприемника по месту использования. Когда пользователь кидает монетку в устройство, она пролетает через несколько датчиков. И если снятые с этих датчиков характеристики монеты совпадают с запрограммированными, то монетоприемник принимает монету.

Рисунок 2 – Программируемый монетоприемник

В данной главе проанализированы поставлены цели и задачи проекта. А также здесь разобраны существующие решения и их особенности. Определены примерные характеристики необходимого устройства.

2 Разработка электрической принципиальной схемы

2.1 Разработка системы управления

Для начала под исходные данные нам необходимо подобрать компоненты, которые смогут обеспечить выполнение требуемого задания.

«Мозгом» схемы послужит микроконтроллер Arduino Nano (Рисунок 3). За счет своих размеров он не будет сильно выделяется и мешать при размещении внутри корпуса копилки.

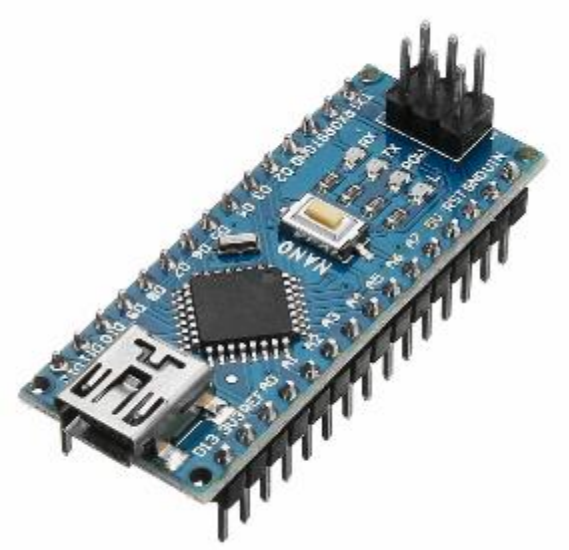


Рисунок 3 – Микроконтроллер Arduino Nano

Выводить всю необходимую информацию будем на ЖК дисплей LCD 1602 с интерфейсом I2C для подключения к микроконтроллеру с наименьшим количеством проводов (Рисунок 4).

Рисунок 4 – ЖК дисплей LCD 1602 с интерфейсом I2C

Определять пролетающие монеты будет монетоприемник, который работает по следующему принципу – в небольшой тоннель, в котором друг напротив друга будут располагаться два датчика – ИК-светодиод и фототранзистор (Рисунок 5). Ширина прохода для пролета монет должна быть чуть больше толщины самой толстой монеты. В нашем случае самая толстая это монета номиналом 10 руб. около 2.5 мм. Делаем с небольшим запасом для учета неровностей тоннеля и получаем ширину порядка 4 мм. Монеты разных номиналов закрывают разную ширину пролета, то есть в итоге мы имеем разный сигнал от различных по размеру монет и за счет этого мы и будем вычислять номинал монеты.

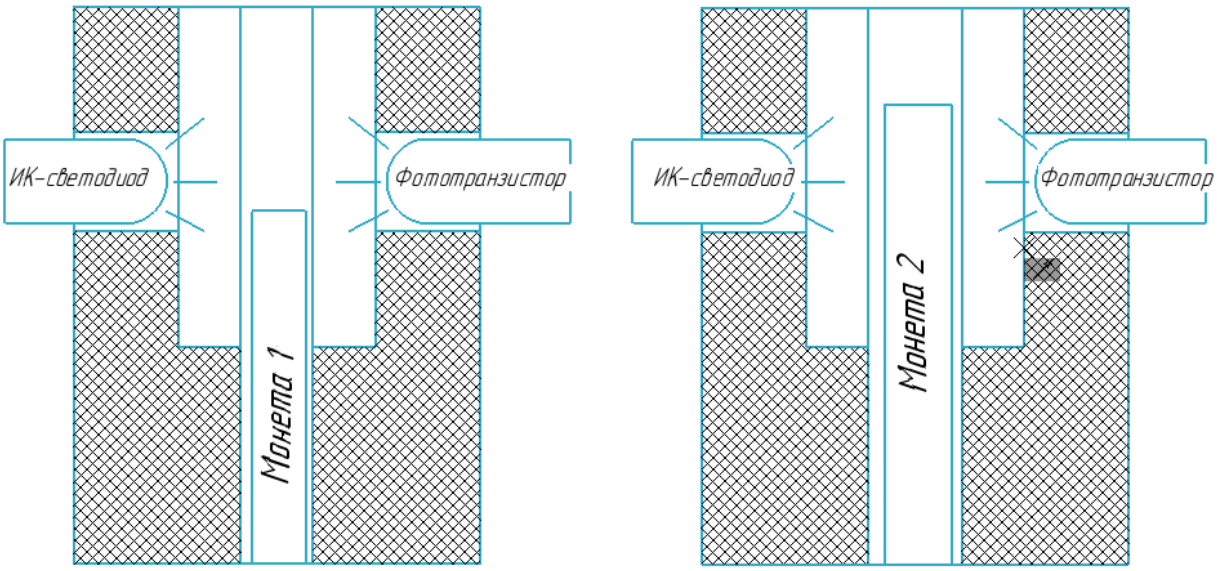


Рисунок 5 - Принцип работы монетоприемника

Также в схему добавим 1 кнопку (Рисунок 6), которая будет отвечать за вход в режим калибровки. Что представляет из себя режим калибровки? Когда мы запускаем первый раз копилку или же мы вносили какие-то изменения в конструкцию монетоприемника, которые отражаются на сигналах от монет или мы будем использовать копилку для других валют, то нам необходимо показать системе монеты, которые ей придется распознавать. Для этого и прописан режим калибровки. Как он работает? Монету каждого номинала кидаем 3 раза в монетоприемник для лучшей точности, чтобы программа посчитала максимальный, средний и наименьший сигнал монет одного номинала. Таким образом, в дальнейшем все пролетающие монеты будут сравниваться с диапазоном, который находится в памяти микроконтроллера, и делать вывод о том, какого номинала пролетевшая монета.

Рисунок 6 – Тактовая кнопка

В схеме будет присутствовать вторая «кнопка», отвечающая за включение режима считывания монет. «Кнопкой» это будет только на схеме, в реальности это будет два контакта (Рисунок 7) на входе в монетоприемник, которые будет замыкать монетка в момент когда ее опускают в щель монетоприемника.

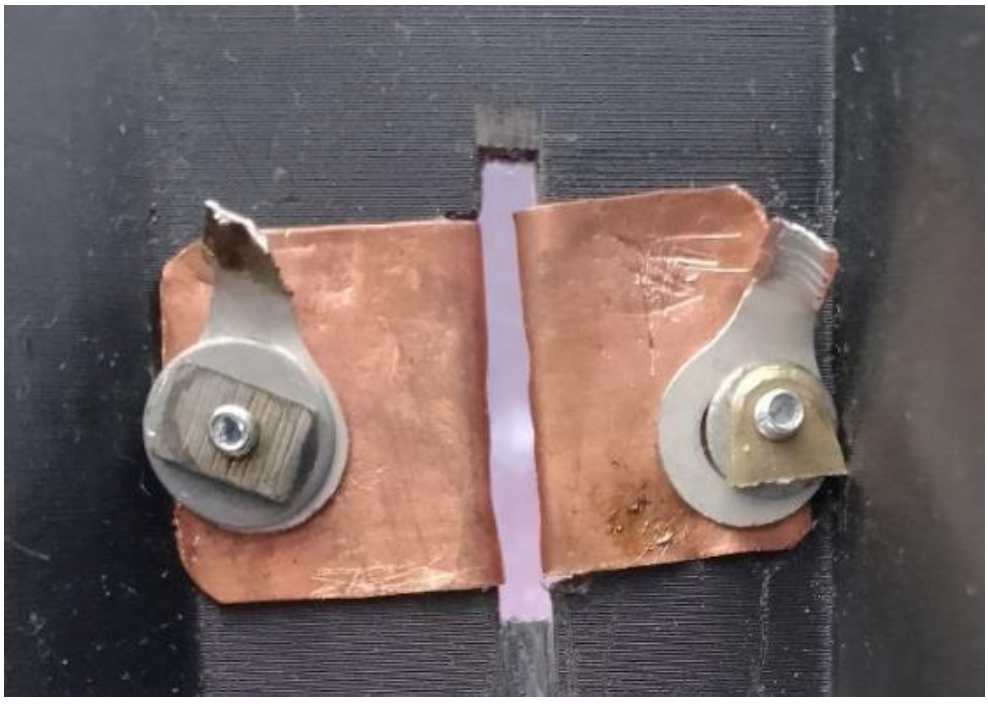


Рисунок 7 – Контакты на входе в монетоприемник

Для работы часов нам понадобится модуль реального времени DS 1302 (Рисунок 8). Этот модуль за счет наличия на плате батарейки CR2032 позволит нам хранить текущую дату и точное время в независимости от наличия питания в микроконтроллере.

Рисунок 8 – Модуль реального времени DS1302

В этот проект я заранее добавляю клавиатуру для будущего расширения функционала данного устройства. Под будущим функционалом я подразумеваю такие опции как: возможность настройки времени без входа в программу, возможность установки будильника. Нам хватит функционала ADKeyboard с 5 кнопками на плате (Рисунок 9).

Рисунок 9 – Модуль клавиатуры ADKeyboard

В качестве источника питания как указано в исходных данных к ВКР выступает аккумуляторная батарея формата 18650. Она будет располагаться на плате Powerbank 18650 (Рисунок 10). Также через эту плату за счет наличия порта для зарядки будет осуществляться и питание от сети.

Рисунок 10 – Powerbank 18650

Таким образом все комплектующие, которые требуются для выполнения задания подобраны. Теперь необходимо разработать схему по которой мы будем соединять эти элементы между собой.

Схема электрическая принципиальная выполнена в программе Компасс 3D v22 (Рисунок 11)

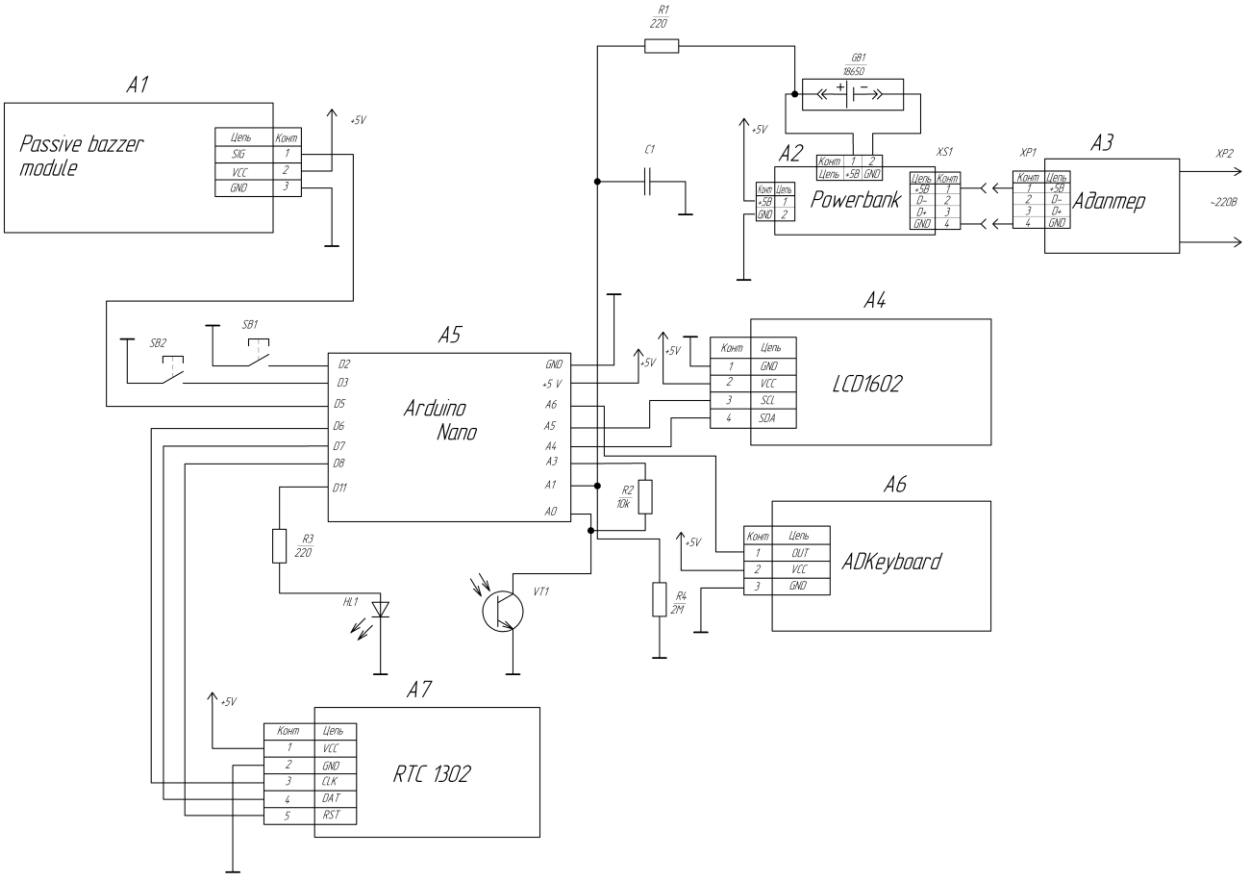


Рисунок 11 – Схема электрическая принципальная

После создания схемы электрической принципиальной можно приступать к написанию кода программы и сборки схемы.

2.2 Написание программы для микроконтроллера

Описание кода программы удобнее будет разделить на блоки с их пояснением в самом коде путем комментирования.

Для начала опишем код монетоприемника, который и будет отвечать за определение монет.

Блок 1 – Здесь прописываем основные настройки программы. В нашем случае будем распознавать 4 номинала монет – 5 рублей, 2 рубля, 10 рублей, 1 рубль. Они расположены в порядке уменьшения диаметра. Также обозначаем

валюту и указываем сколько раз будет кидаться монета одного и того же номинала – 3 считаю наилучшим вариантом, однако чем больше, тем выше точность.

Листинг 1

```
#define coin_amount 4     // число номиналов монет, которые нужно распознать
#define coin_amount_calibration 3  //  число монет одного номинала используемых при калибровке
float coin_value[coin_amount] = { 5.0, 2.0, 10.0, 1.0 };  // номинал монет в порядке уменьшения диаметра
String currency = "RUB"; // валюта
int displ_coin_value_time = 1500; // время отображения на дисплее номинала пролетевшей монеты (миллисекунды)
int stb_time = 15000;     // время бездействия, через которое система перейдет в режим отображения времени
char* print_coin_value[coin_amount] = {"5 pyblei", "2 pybla", "10 ryblei", "1 pybl"}; // текстовый массив номиналов монет в порядке уменьшения диаметра!
```

Блок 2 – обозначаем основные переменные программы.

Листинг 2

```
int coin_signal[coin_amount];     // тут хранится максимальное значение сигнала пролетевшей монеты для каждого размера (номинала) монет
int coin_signal_min[coin_amount];     // тут хранится нижнее значение границы диаппазона сигнала для каждого размера (номинала) монет
int coin_signal_max[coin_amount];     // тут хранится верхнее значение границы диаппазона сигнала для каждого номинала монет
int coin_quantity[coin_amount];  // храним количество монет каждого номинала
int sens_signal, last_sens_signal; //значение уровня сигнала монеты в динамике от фотодатчика analogRead(IRsens)= analogRead(14)
byte empty_signal;              //переменная для уровня фонового сигнала
unsigned long standby_timer; //таймер ожидания (ухода в сон)
unsigned long reset_timer; //таймер очистки памяти, он так же используется для принудительного выхода из калибровки
unsigned long disvalcoin_timer; // таймер, времени вывода на дисплей номинала брошенной монеты.
float summ_money = 0;           // сумма монет в копилке
boolean recogn_flag, sleep_flag = true;     // флажки recogn_flag, sleep_flag = true (true (истина) значение отличное от 0)
boolean coin_flag = false; // coin_flag = false (ЛОЖЬ т.е 0)
```

Блок 3 – подключаем все необходимые для функционирования программы библиотеки. Указываем адрес дисплея.

Листинг 3

```
#include "LowPower.h"
#include "EEPROMex.h"
#include "LCD_1602_RUS.h"
LCD_1602_RUS lcd(0x27, 16, 2); // создать дисплей
```

Блок 4 – обозначаем пины, к которым подключены датчики, кнопки, дисплей.

Листинг 4

```
#define button 2         // кнопка "проснуться":D2, значениев программе
=2
#define calibr_button 3  // скрытая кнопка калибровки и сброса: D3:
значениев программе =3
#define disp_power 12    // питание дисплея: D12, значениев программе
disp_power=12
#define LEDpin 11        // питание ИК диода: D11, значениев программе
LEDpin=11
#define IRpin 17         // питание фотодиода: A3, значениев программе
IRpin=17
#define IRsens 14        // сигнал фотодиода: A0, значениев программе
IRsens=14
```

Блок 5 – в void setup() прописываем те функции программы, которые выполняться один раз при запуске системы. Это и подача питания, считывания фонового сигнала и включение калибровки если нужно пользователю и вывод информации о накоплениях на экран.

Листинг 5

```
void setup() {
  Serial.begin(2400);  // открыть порт для связи с ПК для отладки и
задаем скорость работы монитора порта
  delay(500); // останавливаем выполнение программы на пол секунды
  void(* resetFunc) (void) = 0;//объявляем функцию reset с адресом 0 (для
программного ресет)

  // подтягиваем кнопки
  pinMode(button, INPUT_PULLUP);
  pinMode(calibr_button, INPUT_PULLUP);

  // пины питания как выходы
  pinMode(disp_power, OUTPUT);// питание дисплея
  pinMode(LEDpin, OUTPUT); //  питание ИК диода
  pinMode(IRpin, OUTPUT); // питание фотодиода
  //pinMode(A2, OUTPUT); //  резерв пин 16(A2) питание ещё чего либо
```

```
        // подать питание на дисплей и датчик
        digitalWrite(disp_power, 1); // или digitalWrite(disp_power, HIGH);
питание дисплея
        digitalWrite(LEDpin, 1);   //  digitalWrite(LEDpin, HIGH); питание ИК
диода пин D11,
        digitalWrite(IRpin, 1);    //   digitalWrite(IRpin, HIGH); питание
фотодиода

        // подключить прерывание
        attachInterrupt(0, wake_up, CHANGE);

        empty_signal = analogRead(IRsens);  // считать пустой (опорный/фоновый)
сигнал empty_signal
        Serial.println("1.Void setup.Background signal level: empty_signal = "
+ String(empty_signal)); // 1-й раз вывод уровня фонового сигнала
        // инициализация дисплея
        lcd.init();
        lcd.backlight(); //  Включаем подсветку LCD дисплея
```

Блок 5.1 – Режим калибровки. Включается если при запуске системы зажат кнопка калибровки. Калибровка, как я уже ранее писал, нужна для того, чтобы система сама подстроилась под разные размеры монет и научилась их гарантированно различать. Также в режиме калибровки мы считываем значение фонового сигнала, что очень важно и помогает избежать автоматической калибровки.

Листинг 5.1

```
      if (!digitalRead(calibr_button)) {  // если при запуске нажата кнопка
КАЛИБРОВКА
          for (byte i = 0; i < coin_amount; i++) {
            coin_quantity[i] = EEPROM.readInt(i * 2); // считываем количество
монет каждого номинала, если они были сохранены ранее
          }
          // Вывод на дисплей "Сервис" и информации о процессе калибровки
          lcd.clear();
          lcd.setCursor(5, 0);
          lcd.print(L"Сервис");
          delay(500);
          reset_timer = millis(); //сбросить таймер
          while (1) {                                  // бесконечный цикл
              if (millis() - reset_timer > 10000) {// если кнопка всё ещё
удерживается и прошло 10 секунд,
              // то очистить количество монет в памяти
              for (byte i = 0; i < coin_amount; i++) {
                coin_quantity[i] = 0; // Память ОБНУЛЕНА! Количество монет в
копилке всех номиналов - ноль
```

```
                EEPROM.writeInt(i * 2, 0);  // заполняем все ячейки памяти с
адреса 0 по .... нулями
               }
               lcd.clear();
               lcd.setCursor(1, 0);
               lcd.print(L"Память очищена");
               //Serial.println("METKA 0: The memory is cleared");
               delay(200);
             }
              if (digitalRead(calibr_button)) {    // если отпустили кнопку,
перейти к калибровке
               //Serial.println("METKA 1: Calibration started"); //  калибровка
началась
               lcd.clear();
               lcd.setCursor(3, 0);
               lcd.print(L"Калибровка");
               delay(1500);
               reset_timer = millis(); // сбросить таймер
               break;
             }
           }

           while (1) {
             for (byte i = 0; i < coin_amount; i++) { // последовательный перебор
номиналов монет размер которых калибруется

               for (byte k = 0; k < coin_amount_calibration; k++) { // повторение
калибровки монеты того же номинала (k раз)
                  // вывод на дисплей
                  lcd.setCursor(0, 1); lcd.print(coin_value[i]);  // отобразить
цену монеты, размер которой калибруется
                   lcd.setCursor(6, 1); lcd.print(currency);      // отобразить
валюту
                  lcd.setCursor(13, 1); lcd.print(coin_amount_calibration - k);
// отобразить оставшееся количество попыток калибровки монеты того же номинала

                     empty_signal = analogRead(IRsens);   // считать пустой
(опорный/фоновый) сигнал empty_signal
                      Serial.println("2.Calibration.Background signal level:
empty_signal = " + String(empty_signal)); // 2-й раз вывод уровня фонового
сигнала

                 last_sens_signal = empty_signal;  // значение last_sens_signal
= пустому (опорному/фоновому) сигналу empty_signal

                  while (1) {  // бесконечный цикл сканирования фотодатчика и
анализа полученного значения
                    if (millis() - reset_timer > 60000) resetFunc(); //вызываем
функцию reset по прошествию 60 секунд, если ничего не делали

                   sens_signal = analogRead(IRsens);     // считать фотодатчик
```

```
                    if (sens_signal > last_sens_signal) last_sens_signal =
sens_signal;  // если текущее значение больше предыдущего то запоминаем его
                  if (sens_signal - empty_signal > 60) coin_flag = true;  //
если, значение превысило фоновое на 60, считать что пролетает монета, ИСТИНА
                  if (coin_flag && (abs(sens_signal - empty_signal)) < 25) {
// если, значение упало, вернулось до фонового, то монета точно пролетела
                       coin_signal[i] = last_sens_signal; // зафиксировали
максимальное значение пролетевшей монеты
                   if (k < 1) {
                      coin_signal_max[i] = coin_signal[i];  // если эта была
первая попытка, то ...
                     coin_signal_min[i] = coin_signal[i];
                   }
                  if (coin_signal[i] > coin_signal_max[i]) coin_signal_max[i]
= coin_signal[i]; // фиксируем максимальное значение
                   if (coin_signal[i] < coin_signal_min[i]) coin_signal_min[i]
= coin_signal[i]; // фиксируем минимальное значение
                   coin_quantity[i]++;  // добавляем к общему количеству монет
данного номинала ещё одну

                   coin_flag = false;  // ЛОЖЬ
                     reset_timer = millis(); //сбросить таймер программного
reset для принудительного выхода из калибровки
                   break; // принудительный выход из цикла while (1)
                   } // конец, пролетающая монета отсканирована, возврат в
непрерывный цикл сканирования фотодатчика, ждём пролёт следуещей монеты
                } // скобка цикла while (1) { } непрерывный цикл сканирования
фотодатчика
              } // скобка цикла for (byte k = 0; k < coin_amount_calibration;
k++), повторение калибровки монеты того же номинала (k раз)
            }// скобка цикла for (byte i = 0; i < coin_amount; i++), перебора
номиналов монет размер которых калибруется
            break;// принудительный выход из цикла калибровки while (1)
          } // скобка while (1) { } цикла калибровки
          //---------------Вывод результатов калибровки-------------------
          // для контроля, выводим полученные статистические значение границ
диапазонов распознавания монет
          Serial.println("Statistical boundaries of coin recognition ranges");
          for (byte i = 0; i < coin_amount; i++) {
              Serial.println("coin_signal_max~min[" + String(i) + "]: " +
String(coin_signal_max[i]) + "~" + String(coin_signal_min[i]));
          }
          // Далее, исходя из полученных в цикле калибровки данных идёт расчёт
значений верхних и нижних
          // границ диапазонов для каждого номинала монет, расширенных до их
смыкания
          coin_signal_max[0] = 1023; // верхняя граница монеты с максимальным
диаметром ограничена цифрой 1023
          coin_signal_min[coin_amount - 1] = empty_signal + 15 ; // нижняя
граница монеты с минимальным диаметром на 15 единиц выше фонового значения
          for (byte i = 0; i < coin_amount; i++) {
```

```
            if (i > 0) {
               coin_signal_max[i] = (coin_signal_min[i - 1] - 1); //Верхняя
граница диапазона номинала монеты
            }
             if (i < coin_amount - 1) { // пока переменная last_sens_signal
свободна, используем её для вычеслений границ диапазонов!
              last_sens_signal = (coin_signal_min[i] - coin_signal_max[i + 1])
/ 2; // величина увеличения границы диапазона, округлённая до целого значения
               coin_signal_min[i] = coin_signal_min[i] - last_sens_signal; //
нижняя граница диапазона номинала монеты
            }
          }
           // Записываем как массив данных расчитанные, расширенные границы
диаппазонов распознования монет в память и количество монет
         for (byte i = 0; i < coin_amount; i++) {
           EEPROM.updateInt(i * 2, coin_quantity[i]); // обновляет байт данных
в ячейке за количество монет, если её старое содержимое отличается от нового
           EEPROM.writeInt(20 + i * 2, coin_signal_min[i]); // последовательно
записали минимальное значение за каждую монету в ПАМЯТЬ в ячейку с 40 по ...
           EEPROM.writeInt(40 + i * 2, coin_signal_max[i]); // последовательно
записали максимальное значение за каждую монету в ПАМЯТЬ в ячейку с 60 по ...
         }
         // вывод результата калибровки на дисплей
         for (byte i = 0; i < coin_amount; i++) { // отобразить на дисплее
границы диапазонов распознавания для каждого номинала монет (но эти данные не
из памяти EEPROM!)
           lcd.clear();
           lcd.setCursor(0, 0);
           lcd.print(coin_value[i]); lcd.print(L" "); // номинал монеты
                  lcd.print(coin_signal_max[i]);   lcd.print(L"   ->   ");
lcd.print(coin_signal_min[i]); // и её диапазон
           lcd.setCursor(0, 1);
           if (i < coin_amount - 1) {
             lcd.print(coin_value[i + 1]); lcd.print(L" "); // номинал монеты
                 lcd.print(coin_signal_max[i  +  1]);  lcd.print(L"  ->  ");
lcd.print(coin_signal_min[i + 1]); // и её диапазон
           }
           delay(3000); // с паузой в 3 секунды, последовательно отображаются
результаты калибровки за каждую монету
         }
```

Блок 5.2 – После того как завершилась калибровка (если она была активирована) идет обычный режим работы. Для этого в первую очередь необходимо вывести накопления на экран. Также мы должны из памяти считать значения верхнего и нижнего диапазонов для каждого номинала монет.

Листинг 5.2

```
        for (byte i = 0; i < coin_amount; i++) {
           coin_quantity[i] = EEPROM.readInt(i * 2); // считывает количество монет каждого номинала
          coin_signal_min[i] = EEPROM.readInt(20 + i * 2); // считывает значение нижней границы диапазона
           coin_signal_max[i] = EEPROM.readInt(40 + i * 2); // считывает зачение верхней границы диапазона
            summ_money += coin_quantity[i] * coin_value[i];  // считаем сумму монет, как произведение количества монет на их номинал
        }
         Serial.println("Expanded coin recognition boundaries read from memory"); // Расширенные границы распознавания монет, считанные из памяти
        for (byte i = 0; i < coin_amount; i++) {
            Serial.println("coin_signal_max~min[" + String(i) + "]: " + String(coin_signal_max[i]) + "~" + String(coin_signal_min[i]));
        }
        Serial.println("3.Void setup.Background signal level: empty_signal = " + String(empty_signal)); // 3-й вывод уровня фонового сигнала
        Serial.println("Amount of money: summ_money = " + String(summ_money)); // вывод суммы денег в копилке
        standby_timer = millis();  // обнулить таймер ухода в сон
      }
```

Блок 6 – Основной цикл программы. В этом цикле программа работают большую часть времени. Здесь мы каждый раз опрашиваем датчики и снимаем с них сигнал на случай если он менялся, то есть если пролетала монета. Затем мы сравниваем все полученный сигнал с теми диапазонами, что получились при калибровке и принимаем решение о принадлежности монеты к какому-либо номиналу.

Листинг 6

```
      void loop() {
        if (sleep_flag) {
          delay(500);// пауза пол секунды

           // если проснулись после сна, инициализировать дисплей и вывести текст, сумму, валюту и заряд батареи
          lcd.init(); // инициализация дисплея
          lcd.clear();
          lcd.setCursor(0, 0); lcd.print(L" Умная копилка");
          lcd.setCursor(0, 1); lcd.print(summ_money); // если нужно отобразить целые числа, то lcd.print(summ_money,0) или ((int)summ_money)
          lcd.setCursor(13, 1); lcd.print(currency);

          empty_signal = analogRead(IRsens); // считать датчик (считать пустой (опорный/фоновый) сигнал empty_signal)
```

```
          Serial.println("4.Void loop.Background signal level: empty_signal =
" + String(empty_signal)); // 4-й вывод уровня фонового сигнала
          sleep_flag = false;
        }
         last_sens_signal = empty_signal;  // максимальное значение сигнала
последней пролетевшей монетки last_sens_signal сбрасывается до уровня фона
empty_signal
        // Далее работаем в бесконечном цикле
        // While будет работать в цикле непрерывного сканирования до тех пор,
пока время таймера ухода в сон не вышло и break
        while (1) {
          // далее такой же алгоритм, как при калибровке
          sens_signal = analogRead(IRsens);  // считать датчик
              if   (sens_signal   >   last_sens_signal)   last_sens_signal   =
sens_signal;  // если текущее значение больше предыдущего
           if (sens_signal - empty_signal > 60) coin_flag = true;  // если
значение упало почти до "пустого", считать что монета улетела
         if (coin_flag && (abs(sens_signal - empty_signal)) < 25) {    // если
монета точно улетела
            recogn_flag = false;  // флажок ошибки, пока что не используется
             // в общем нашли максимум для пролетевшей монетки, записали в
last_sens_signal
             // дальше в программе он сбросится до уровня пустого/фонового
сигнала empty_signal
             // далее начинаем сравнивать сигнал с диапазонами за каждую из
монет, хранящимися в памяти

            for (byte i = 0; i < coin_amount; i++) {
              if ( last_sens_signal >= coin_signal_min[i] && last_sens_signal
<= coin_signal_max[i]) {  // !!! и вот тут если сигнал попадает в диапазон, то
считаем монетку распознанной
                summ_money += coin_value[i];// к сумме тупо прибавляем цену
монетки (да, сумма считается двумя разными способами. При старте системы суммой
всех монет, а тут прибавление
                  coin_quantity[i]++;  // для распознанного номера монетки
прибавляем количество

               //  вывести на дисплей
               lcd.clear();
              lcd.setCursor(4, 0); lcd.print(print_coin_value[i]); // Номинал
распознанной монеты прописью
             // lcd.setCursor(0, 0); lcd.print(coin_value[i], 2);  // Номинал
распознанной монеты цифрой
                lcd.setCursor(0, 1); lcd.print(summ_money);//  вывести сумму
монет
              lcd.setCursor(13, 1); lcd.print(currency);// отобразить валюту
                                                  //lcd.setCursor(12,       1);
lcd.print(last_sens_signal);lcd.print(L"  "); // уровень сигнала пролетевшей
монеты (значения от фона до 1023)
```

```
                  // выводим на дисплей номинал распознаной монеты русскими
буквами
                if (i == 0 && print_coin_value[i] == "RU") {
                  lcd.setCursor(0, 0);
                  lcd.print(L"    5 рублей     ");
                }
                if (i == 1 && print_coin_value[i] == "RU") {
                  lcd.setCursor(0, 0);
                  lcd.print(L"   2 рубля    ");
                }
                if (i == 2 && print_coin_value[i] == "RU") {
                  lcd.setCursor(0, 0);
                  lcd.print(L"    10 рублей     ");
                }
                if (i == 3 && print_coin_value[i] == "RU") {
                  lcd.setCursor(0, 0);
                  lcd.print(L"   1 рубль    ");
                }

                recogn_flag = true;  // ИСТИНА значение отличное от 0
                  break;// Break используется для принудительного выхода из
цикла, for (byte i = 0; i < coin_amount; i++). Монета опознана, не надо
перебирать другие границы.
              }  //  Монета попала в диапазон, распознана, посчитана и выведена
на дисплей. Break выходим из цикла for (byte i = 0; i < coin_amount; i++)
            } // конец цикла for (byte i = 0; i < coin_amount; i++) перебора и
сравнения сигнала пролетевшей монетой с хранящимися в памяти

            coin_flag = false;
            standby_timer = millis();  // сбросить таймер,  millis() количество
миллисекунд с момента начала выполнения программы
            break;
          }
           // номинал последней пролетевшей монеты отображается на время
продолжительностью displ_coin_value_time милисекунд,
          // если это время вышло, то номинал монеты пролетевшей перестаёт
отображаться и выводится "Умная копилка"
          if (millis() - standby_timer >= displ_coin_value_time && millis() -
standby_timer < displ_coin_value_time + 4) {
            lcd.clear();
            lcd.setCursor(0, 0); lcd.print(L" Умная копилка");
            lcd.setCursor(0, 1); lcd.print(summ_money);
            lcd.setCursor(13, 1); lcd.print(currency);
          }

          // если ничего не делали, время таймера вышло, спим (stb_time время
бездействия, через которое система уйдёт в сон )
          if (millis() - standby_timer > stb_time) {
            good_night();
            break;
          }
```

```
          while (!digitalRead(button)) { // нажата кнопка "проснуться" (или
монетка замыкает контакты)
           if (millis() - standby_timer > 5000) { // если контакты "проснуться"
замкнуты > 5 секунд, то вывод статистики
              lcd.clear();
               //Вывод номиналов монет сверху и вывод статистики количества
монет снизу !
             for (byte i = 0; i < 4; i++) { // отобразить на дисплее статистику
для первых 4-х монет
               lcd.setCursor(i * 4, 0); lcd.print(coin_value[i], 1); // сверху
номенал монеты (округлен до десятых)
               lcd.setCursor(i * 4, 1); lcd.print(coin_quantity[i]); // снизу
их количество
              }
              delay(5000);
              if (coin_amount > 4) { // если число номиналов монет больше чем
4, вывод в два этапа
               lcd.clear();
                for (byte i = 4; i < coin_amount; i++) { // отобразить на
дисплее статистику для остальных монет
                  lcd.setCursor(i * 5 - 20, 0); lcd.print(coin_value[i], 2);
// сверху номенал монеты (округлен до десятых)
                  lcd.setCursor(i * 5 - 20, 1); lcd.print(coin_quantity[i]);
// снизу их количество
                }
              }
              delay(5000);
              standby_timer = millis() ;  //сбросить таймер
            }
          }

        }
      }
```

Код блока часов, а также модуля пассивного зуммера уже позже встраиваются в код счетчика монет, т.к. гораздо меньший объем и второстепенное значение для работы. Полный код всей программы размещен в Приложении А. Теперь же можно приступать к сборке и тестированию работоспособности программы.

В данной главе подобраны элементы для схемы с учетом требований, предъявляемых заданием. Также составлена схема электрическая принципиальная и написан код программы.

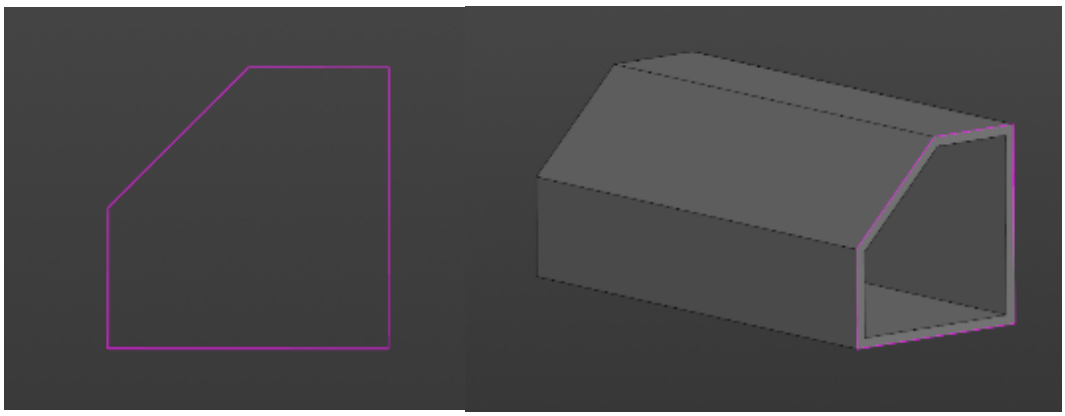
3 Практическая реализация проекта

3.1 Сборка модели

3.1.1 Моделирование корпуса копилки

Так как в данной ВКР разрабатывается практичное устройство, которое подошло бы для повседневного использования, то у нее должен быть и практичный корпус, который к тому же неплохо смотрелся бы на рабочем столе в офисе или дома. Начнем построение модели в программе Компасс-3Д v.22.

Так как это копилка, то в нее должно помещаться много денег, соответственно делаем ее большой, чтобы вмещать крупные суммы мелочи. Габариты копилки у меня получились следующими: высота – 100 мм; ширина – 100 мм; длина – 250 мм. Также считаю правильным сделать одну грань корпуса под наклоном, т.к. предполагается, что на дисплей будут смотреть сверх вниз под некоторым углом сидя за столом. Получается эскиз (*Рисунок 12*а), который необходимо выдавить на те самые 250 мм (Рисунок 12 б), что я обозначил выше. Толщина стенок при этом у меня предполагается 5 мм, что на самом деле является чрезмерной и можно было бы взять гораздо меньше. Однако такая толщина обеспечивает копилке повышенную прочность, поэтому оставляем и ее.



а) б)

Рисунок 12 - Эскиз корпуса копилки (а) и результат выдавливания эскиза (б)

Добавляем к копилке заднюю стенку (Рисунок 13), чтобы монеты не вываливались просто так.

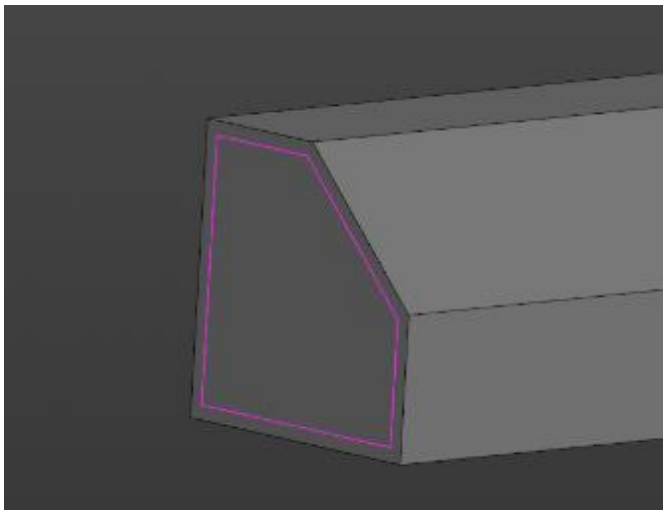
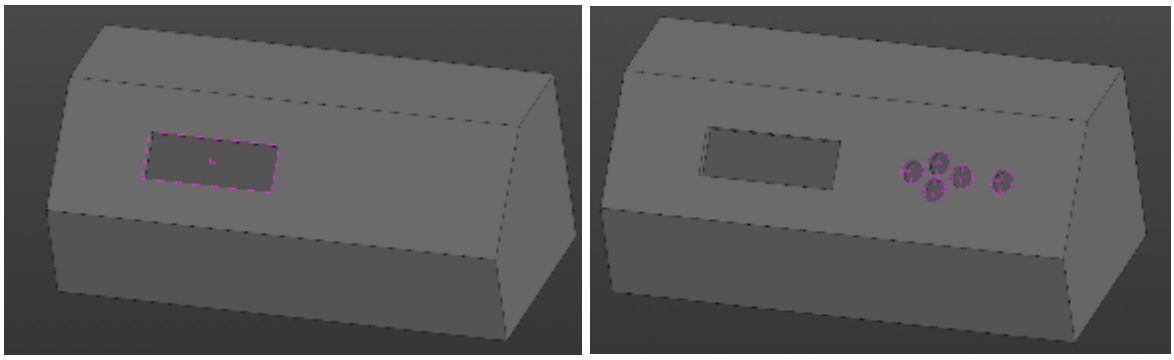


Рисунок 13 - Задняя стенка устройства

Теперь начнем вырезать в корпусе функциональные отверстия под устройства схемы. Первым вырежем отверстие под ЖК дисплей LCD 1602 (Рисунок 14а). Размеры все следует снимать предельно точно, чтобы в дальнейшем когда корпус будет напечатан на 3Д принтере детали встали в корпус как и положено. При этом следует сразу же закладывать допуск на усадку пластика. Для того пластика, которым будем печатать мы увеличение составит примерно 0,3%, то есть дополнительно в отверстия надо заложить с

запасом по 0,5 мм. Тоже самое делаем с модулем клавиатуры и отверстиями под клавиши (Рисунок 14б). Здесь еще очень важно соблюсти межосевое расстояние, чтобы кнопки точно попали в свои отверстия.



а) б)

Рисунок 14 - Отверстие под ЖК дисплей LCD 1602 (а) и под клавиши ADKeyboard (б)

Теперь следует сделать отверстие под монеты, куда их будем кидать. Диаметр самой большой монеты 25 мм (5 руб.), а толщина самой толстой (10 руб.) – 2 мм. Делаем слегка с запасом + допуск на усадку и получаем размеры 30 в длину и 3,5 мм в ширину. Вырезаем получившийся эскиз (Рисунок 15).

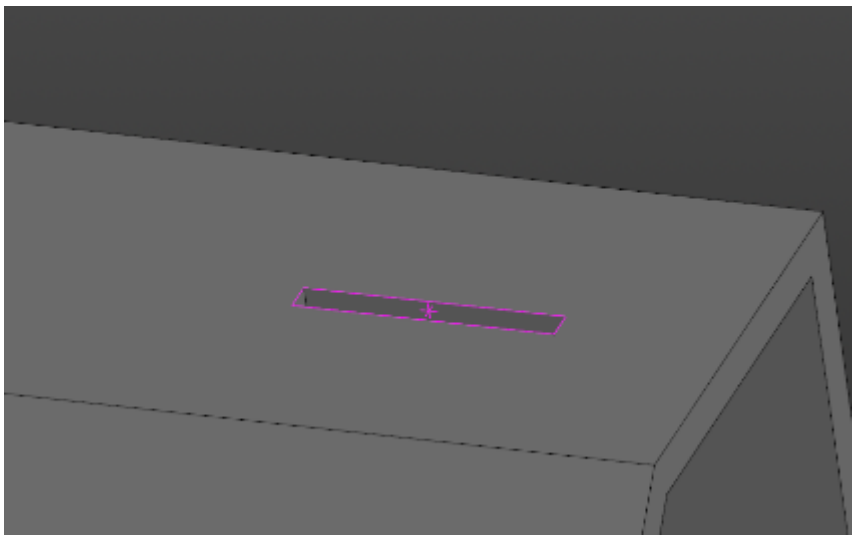


Рисунок 15 - Отверстие под монетоприемник

Следующим шагом необходимо сделать вырез под дверку, через которую в дальнейшем безопасно для самой копилки будет происходить опустошение ее запасов. Для этого в первую очередь убираем стенку (Рисунок 16).

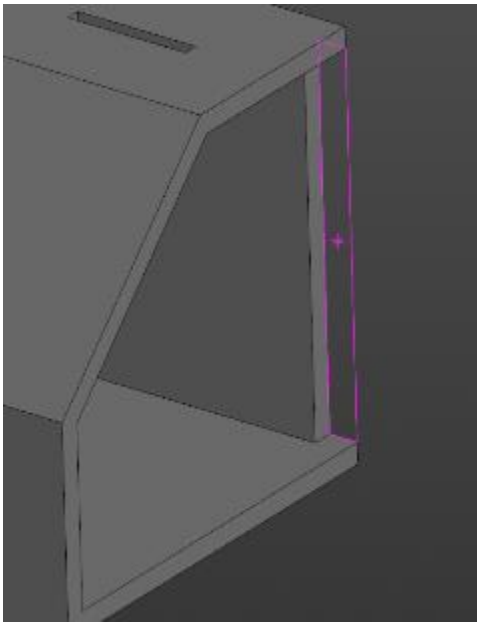
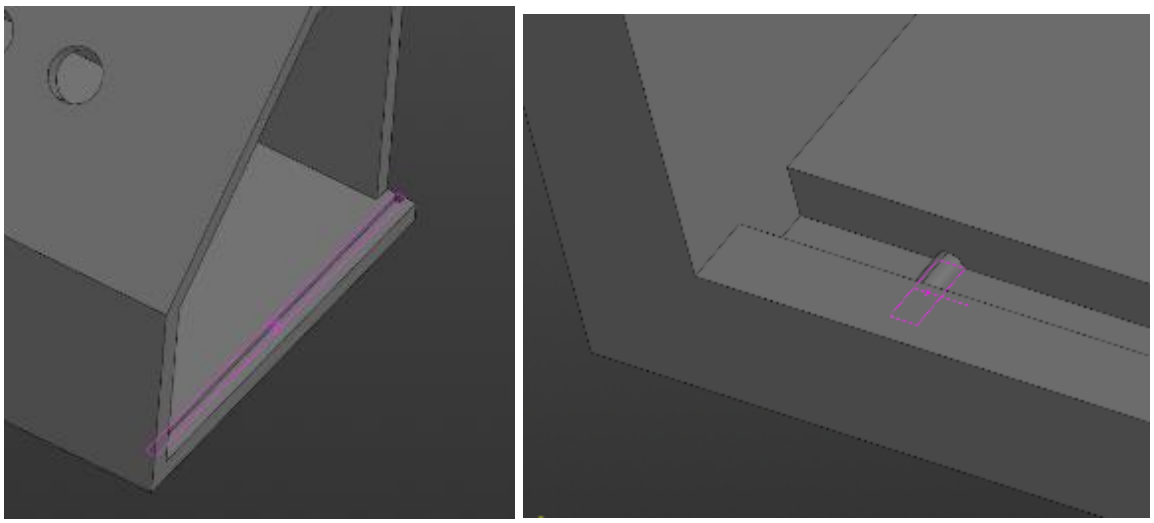


Рисунок 16 - Заготовка под крышку устройства

Теперь необходимо сделать направляющие, по которым крышка будет спокойно двигаться. Делаем пазы по 2 мм в ширину и 2 мм в толщину сверху и снизу корпуса. Вырезаем и получаем пазы (Рисунок 17а). Также необходимо сделать так, чтобы крышка плотно фиксировалась и не открывалась сама по себе. Для этого сделаем бугорок, отверстие под который потом вырежем в другой детали, крышке корпуса. У конца паза делаем небольшой выступ и скругляем его (Рисунок 17б).



а) б)

Рисунок 17 - Пазы для крышки (а) и выступ у конца паза (б)

Наконец делаем отверстие под шнур зарядки. Повербанк расположим в конце корпуса и приклеем его к верху корпуса, чтобы случайные монеты не попали в него. Поэтому отверстие будем делать с обратной стороны в правом верхнем углу. На повербанке для зарядки используется Micro USB и чтобы штекер влез в отверстие нам понадобится отверстие примерно 10мм в длину и 8 мм в высоту (Рисунок 18).

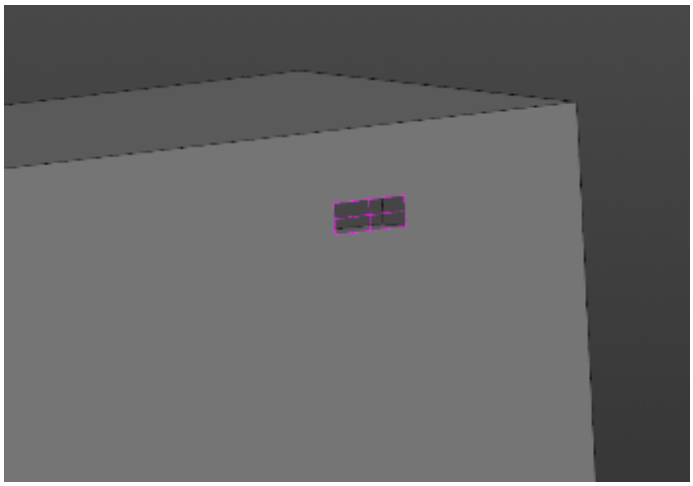


Рисунок 18 - Отверстие под штекер зарядки для повербанка

Еще один важный элемент, который надо продумать это монтажное отверстие, через которое будет приклеиваться вся схема внутри корпуса. Отступаем от краев по 15 мм и делаем прямоугольный вырез шириной 2 мм (Рисунок 19). Позже оставшуюся часть мы приклеим ко дну, когда схема будет собрана.

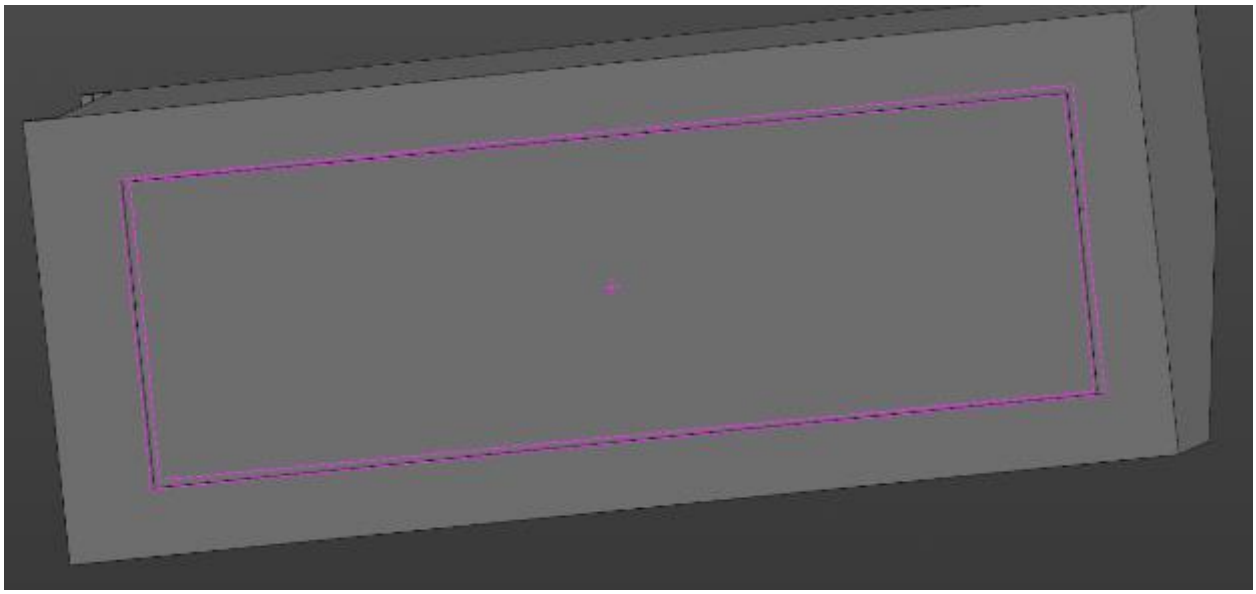


Рисунок 19 – Вырез для монтажа схемы

Остается только придать эстетичности корпусу и сделать небольшие скругления по краям (Рисунок 20).

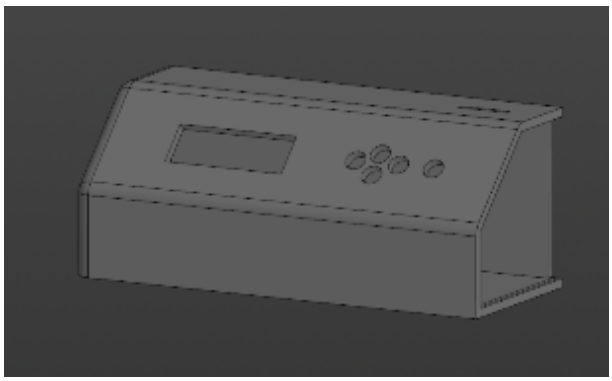


Рисунок 20 - Итоговая модель корпуса копилки

Модель корпуса копилки готова. Ее остается перевести в формат .stp и отправлять на печать. Переходим к моделированию крышки корпуса.

3.1.2 Моделирование крышки корпуса

Крышка должна вписаться в те отверстия и пазы, что мы сделали для нее при проектировании корпуса. Поэтому первый эскиз с размерами копируем из корпуса и вытягиваем получившуюся модель на 5 мм. Она будет такой же толщины, как и остальной корпус (Рисунок 21).

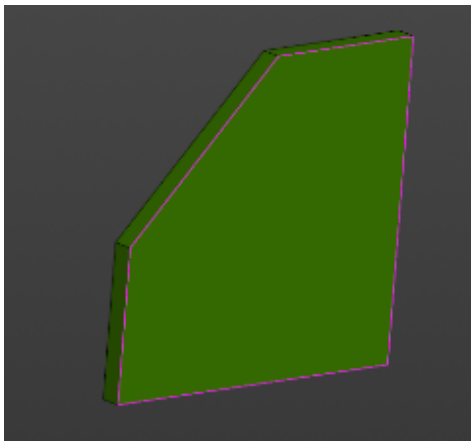
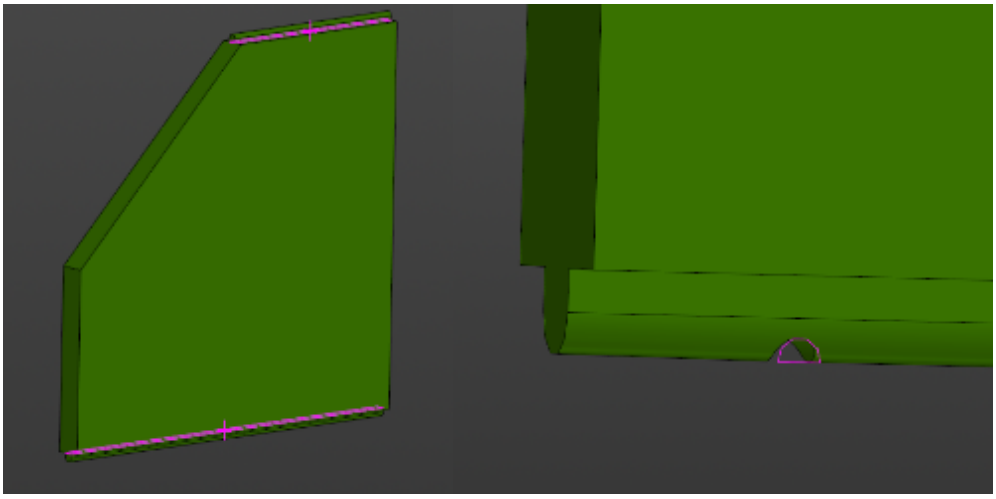


Рисунок 21 - Заготовка крышки корпуса

Теперь к данной заготовке нам необходимо добавить выступы под пазы и немного их скруглить, чтобы они нормально ходили в пазах (Рисунок 22а). Также в нижнем пазе делаем небольшой вырез в заранее намеченном месте под бугорок в пазе корпуса (Рисунок 22б).



а) б)

Рисунок 22 - Выступы под пазы со скруглением (а) и вырез под бугорок (б)

Последним штрихом на середину лицевой грани добавляем три небольших выступа нужных для того, чтобы удобнее было отодвигать крышку в сторону (Рисунок 23).

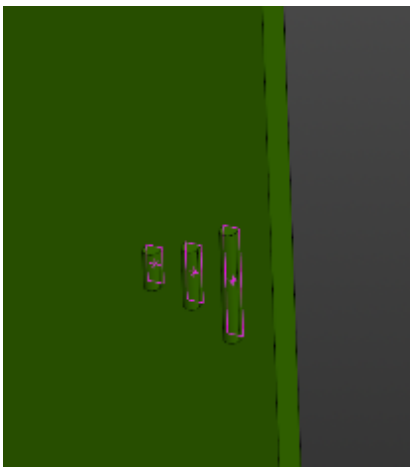
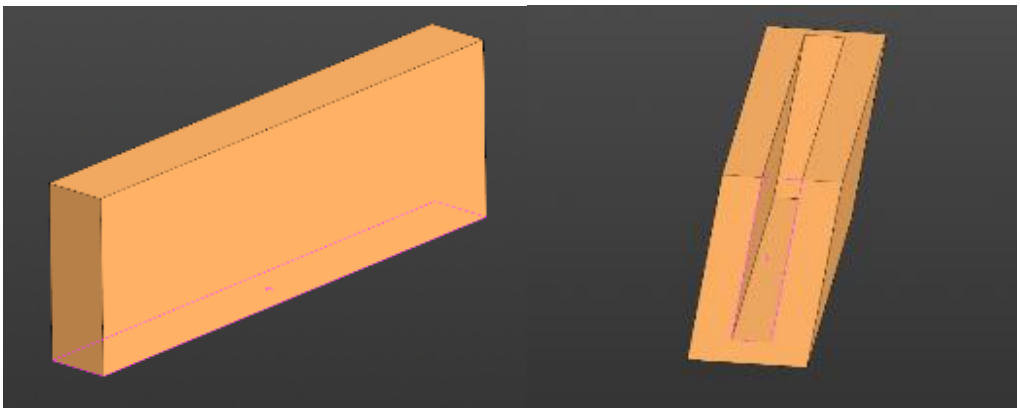


Рисунок 23 – Три выступа на крышке

Крышка корпуса готова, переводим ее в формат .stp и отправляем на печать.

3.1.3 Моделирование монетоприемника

Теперь переходим к моделированию той части устройства, которая будет ответственная за распознавание монет. Для начала нам надо определиться с габаритами детали. Самая большая монета, которая предполагается для распознавания это 5 рублей, диаметр которой 24.5 мм. Нам необходимо чтобы самая большая монета пролетала в монетоприемнике без каких-либо проблем. Поэтому высоту детали сделаем в 30 мм. Щель монетоприемника будет такой же, как и в корпусе – 3.5 мм в ширину. Плюс к этому числу надо прибавить толщину стенок монетоприемника – по 4 мм с каждой стороны. Итого получим 11.5 мм в ширину. В длину возьмем 100 мм, этого хватит с лихвой (Рисунок 24а). Теперь сделаем в этой заготовке щель для монет. Толщина щели, как уже обозначал выше – 3.5мм. От нижнего края заготовки отступим 4 мм и начертим эскиз, по которому уже и вырежем щель (Рисунок 24б).



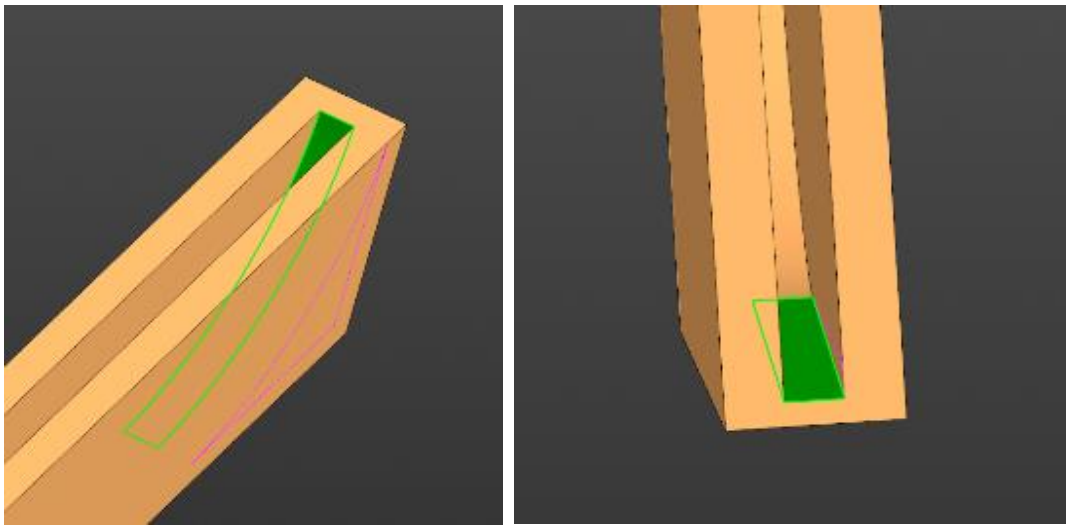
а) б)

Рисунок 24 – Элемент выдавливания (а) и вырез щели под монеты (б)

Чтобы монеты не застревали просто так в монетоприемники, а скатывались сделаем следующие два элемента: горку и уклон.

Элемент горка представляет собой треугольник с дугой радиусом 30 мм (Рисунок 25а). В дополнение к горке как я уже писал выше сделаем небольшой

уклон в 2° (Рисунок 25б).



а) б)

Рисунок 25 – Элемент выдавливания и эскиз горки (а) и эскиз и вырезание уклона (б)

Теперь по бокам сделаем уплотнения, которые нам понадобятся, чтобы отодвинуть датчики от щели и сделать свет более рассеянным Рисунок 26.

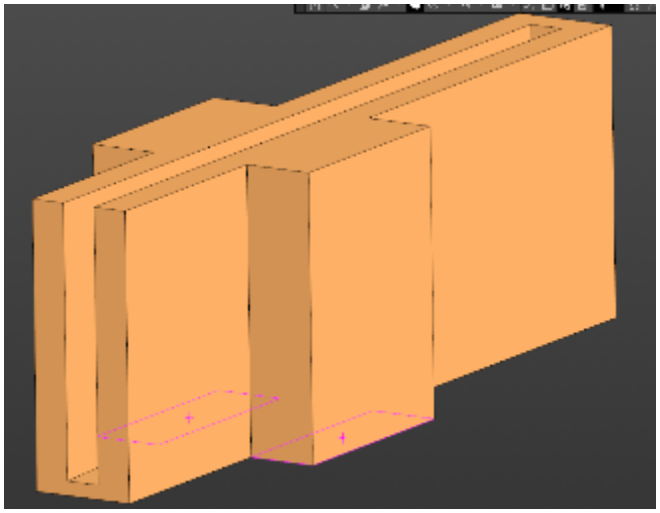


Рисунок 26 – Уплотнения с эскизом для датчиков

Теперь в этом уплотнение нам надо сделать небольшое окошечко. Размеры окошка возьмем 10х10 мм. Позиционируем окошко по длине по середине уплотнения. А по высоте отступаем от верхнего края монетоприемника 4 мм. Так это окошко по высоте охватит в достаточной степени все монеты от 5 до 1 рубля. В это окошко в дальнейшем вставится небольшой кусочек пластика, который мы зашкурим для лучшего рассеивания света. Такое окошко делаем с обеих сторон (Рисунок 27а). Остается только в этом окошке сделать вырез, через который и будет видна сама щель. Вырез по ширине нужен небольшой, достаточно будет 5 мм, под диаметр светодиода (Рисунок 27б).

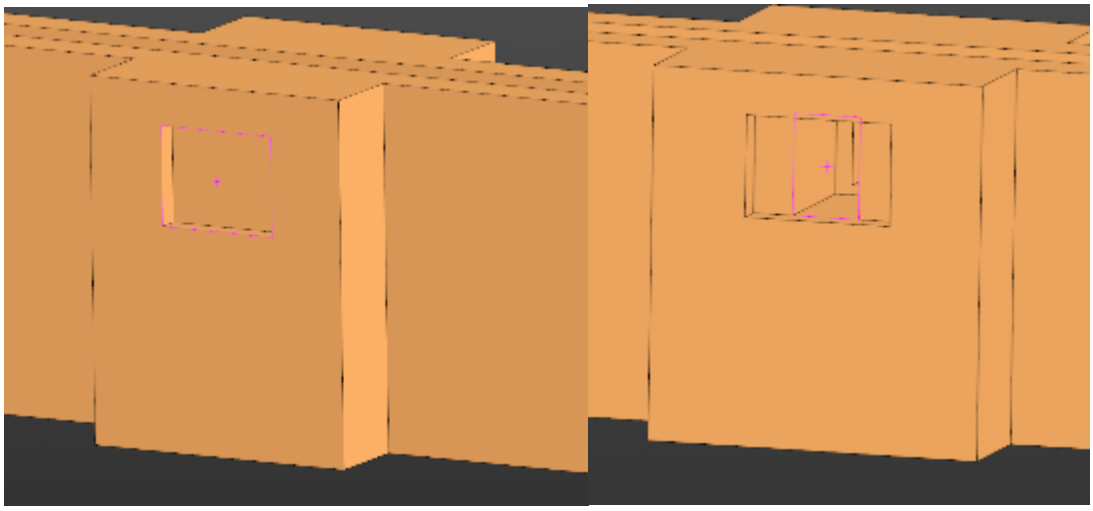


Рисунок 27 – Окошко для вставки пластика (а) и отверстие под датчики (б)

Все необходимые элементы созданы. Теперь необходимо эти 3Д модели отправить на печать на 3Д принтере. Элементы мне напечатает другой человек, поэтому переходим к процессу сборки.

3.1.4 Сборка устройства.

Сборку начинаем с того, что в спаиваем между друг другом все элементы (Рисунок 28).

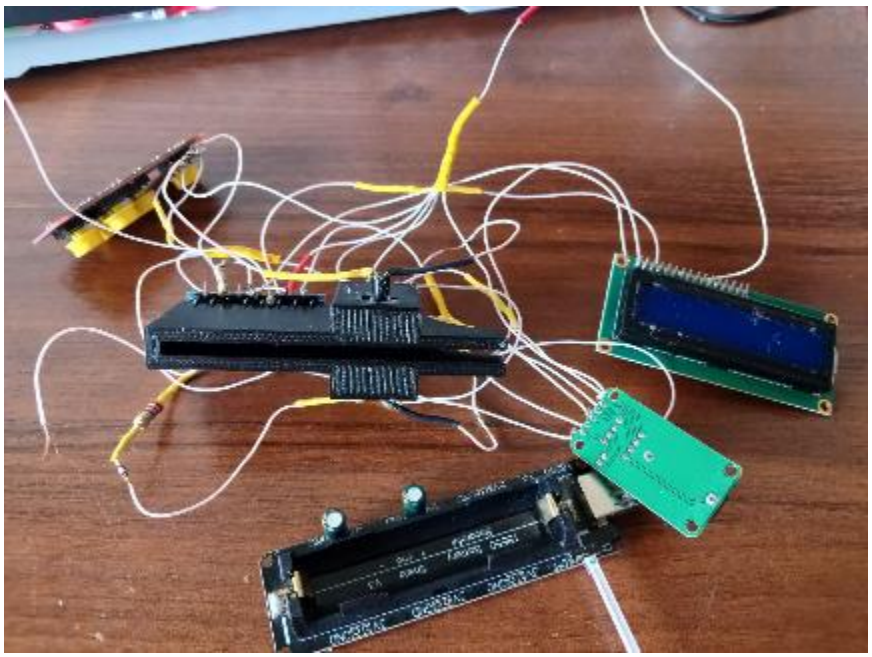


Рисунок 28 – Спаянная схема

После этого спаянную схему приклеиваем внутрь корпуса на термоклей. Там же сразу приклеиваем излишки проводов к верху корпуса, чтобы они не болтались и по ним случайно не задевали монетки. В самом конце припаиваем провода на контакты под кнопку «Проснуться» (Рисунок 29).

Рисунок 29 – Приклеенные элементы внутри корпуса

После того как все элементы схемы приклеены внутри корпуса, остается только приклеить нижнюю часть, которую использовали для монтажа и поставить корпус на ножки. Модель готова (Рисунок 30).

Рисунок 30 – Готовая модель электронного счетчика монет с часами

Теперь, когда модель готова, перейдем к ее отладке.

3.2 Отладка модели

В данном подразделе отражены трудности, с которыми я столкнулся во время сбора схемы и ее калибровки и то как я их решал:

- На дисплей ничего не выводилось.

Решение: Отрегулировать потенциометр сзади адаптера I2C для увеличения яркости.

- На дисплее вместо букв и чисел выводились квадраты (Рисунок 31).

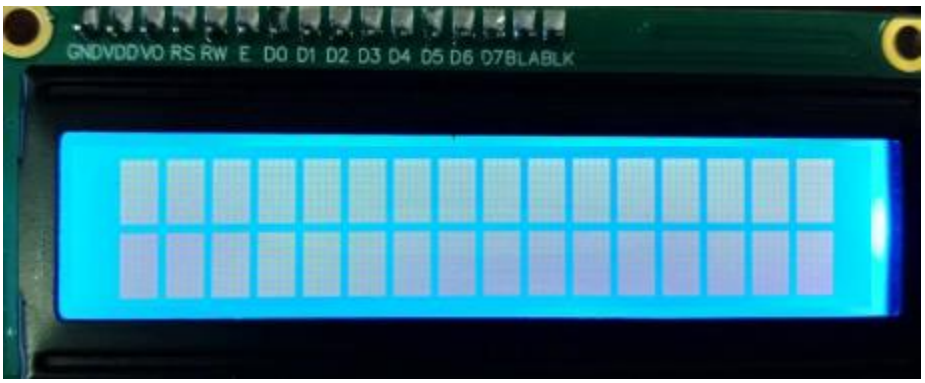


Рисунок 31 - Отображение квадратов на дисплее вместо букв

Решение: Сменить адрес дисплея с 0x3f на 0x27 (Рисунок 32).

Рисунок 32 - Отображение текста на дисплее, после изменения адреса

- При входе в режим калибровки датчик не калибровался корректно потому, что программа воспринимала фоновый уровень сигнала за пролетающие монеты. Таким высоким был этот уровень.

Решение: Написан блок программы для учета сигнала фона. Так теперь при подаче питания, программа считывает с датчика какой на нем уровень сигнала и записывает его в специальную переменную. Это исключает возможность «автокалибровки».

3.3 Экспериментальные исследования

После того как мы собрали и отладили модель самое время провести экспериментальные исследования. В данном подразделе будет показано то, как датчики видят монеты разных номиналов.

По результатам исследования (Рисунок 33) можно сделать вывод, что монетоприемник работает корректно и четко может различать монеты всех ходовых номиналов российского рубля.

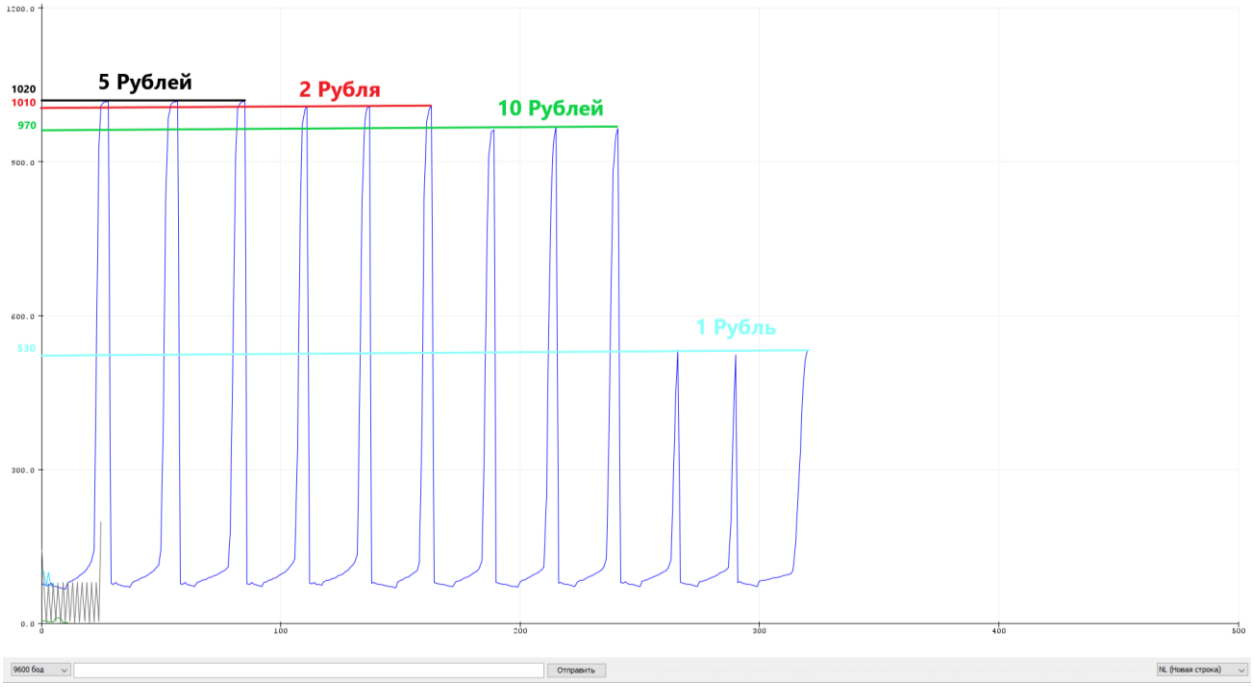


Рисунок 33 – Результаты снятые с плоттера

3.4 Описание лабораторного стенда

В результате работы над данным проектом было получено готовое изделие. Оно останется на кафедре «Промышленная электроника», где будет демонстрироваться на различных выставках и другим студентам с кафедры в качестве примера на лабораторных работах. С учетом всего вышесказанного следует подготовить инструкцию по демонстрации стенда. Полная инструкция

с описанием стенда приведена в Приложении Б. Здесь приведем лишь небольшую инструкцию к стенду.

Инструкция по демонстрации стенда:

1. Установить стенд на столе.

2. На задней стороне стенда переключить рычажок On/Off в верхней правой части корпуса. Если дисплей не загорелся подключить стенд шнуром micro-USB к источнику питания.

3. Режим считывания монет включается поднесением монеты в отверстие монетоприемника и замыканием монетой пластин в отверстие. После чего в отверстие можно кидать монетки для считывания. Этот режим действует 3 секунды, после чего снова выводится время.

4. Режим вывода на дисплей статистики включается удержанием монеты в контактах в течении 3 секунд. Он выключается при условии что монета уже не замыкает контакты и прошло 3 секунды.

В данной главе были смоделированы все составные части копилки. Они же были напечатаны на 3Д принтере. Устройство было собрано и протестировано. В ходе тестирования доказана работоспособность монетоприемника.

Заключение

В ходе выполнения данной выпускной квалификационной работы был разработан электронный счетчик монет, сочетающий в себе функции копилки и часов.

Для достижения цели работы было сделано следующее:

- проанализированы существующие решения;
- подобраны компоненты для устройства;
- разработана схема электрическая принципиальная;
- написан код программы для работы устройства;
- спаяны элементы в одну схему;
- смоделирован и распечатан корпуса устройства;
- собраны элементы в корпусе
- отлажена работа схемы;

Для распознавания монет была использована связка из ИК-светодиода и фототранзистора, что позволило точным образом отличать монеты разных номиналов.

При моделировании устройства опытным путем был подобран наиболее оптимальный вариант монетоприемника, который позволил безошибочно отличать монеты разных диаметров.

При разработке программного обеспечения были решены вопросы совмещения модулей для разных задач в одном устройстве. Это и позволило создать устройство, которое сочетает в себе функции копилки и часов. Также был решен вопрос большого фонового сигнала, который не давал корректно откалибровать систему.

В конечном итоге получилось функциональное устройство, которое можно использовать как дома, так и в офисе на рабочем столе.

Список используемых источников

1. Ардуино и клавиатуры / [Электронный ресурс] // Habr : [сайт]. — URL: https://habr.com/ru/articles/460409/ (дата обращения: 25.04.2024).

2. Arduino. LCD 1602. I2C и другие варианты подключения. / [Электронный ресурс] // MicroTechnics : [сайт]. — URL: https://microtechnics.ru/arduino-lcd-1602-i2c-i-drugie-varianty-podklyucheniya/ (дата обращения: 04.03.2024).

3. Arduino Nano / [Электронный ресурс] // Arduino : [сайт]. — URL: https://arduino.ru/Hardware/ArduinoBoardNano (дата обращения: 20.02.2024).

4. Гавер А. Arduino и зуммер / Гавер А. [Электронный ресурс] // GyverKIT : [сайт]. — URL: https://kit.alexgyver.ru/tutorials/buzzer/ (дата обращения: 23.03.2024).

5. Гавер А. Копилка со счетчиком монет / Гавер А. [Электронный ресурс] // GitHub : [сайт]. — URL: https://github.com/AlexGyver/MoneyBox_counter (дата обращения: 27.03.2024).

6. Монетоприемники NRI / [Электронный ресурс] // Игротехникс : [сайт]. — URL: https://www.igrotechnics.ru/coinacceptors/monetopriemniki-nri.html (дата обращения: 21.02.2024).

7. Назаров, В. В. ОСНОВЫ ЭЛЕКТРОМОНТАЖА [Текст] / В. В. Назаров — 1-е изд.. — Москва: МГТУ им. Н.Э. Баумана, 2019 — 75 с.

8. Олег Евсегнеев Ардуино: оптический датчик препятствия / Олег Евсегнеев [Электронный ресурс] // Класс Робототехники : [сайт]. — URL: https://robotclass.ru/tutorials/arduino-proximity-sensor/ (дата обращения: 15.05.2024).

9. Основы 3D-моделирования для 3D-печати / [Электронный ресурс] // Habr : [сайт]. — URL: https://habr.com/ru/articles/417605/ (дата обращения: 20.05.2024).

10. Пищалка – пьезодинамик Ардуино / [Электронный ресурс] // ArduinoMaster : [сайт]. — URL: https://arduinomaster.ru/uroki-arduino/pishhalka-pezodinamik-arduino/ (дата обращения: 14.05.2024).

11. Подключение DS1302 к Arduino / [Электронный ресурс] // RobotChip

: [сайт]. — URL: https://robotchip.ru/podklyuchenie-ds1302-k-arduino/ (дата обращения: 21.02.2024).

12. Ревич, Ю. В. Программирование микроконтроллеров AVR: от Arduino к ассемблеру [Текст] / Ю. В. Ревич — 1-е изд.. — Санкт-Петербург: БХВ-Петербург, 2020 — 512 с.

13. Сворень, Р. А. Электроника шаг за шагом [Текст] / Р. А. Сворень — 3-е изд.. — Москва: ДМК Пресс, 2020 — 512 с.

14. Структура программы на языке C++ для Arduino / [Электронный ресурс] // Амперка : [сайт]. — URL: http://wiki.amperka.ru/программирование:структура-программы-для-arduino (дата обращения: 12.03.2024).

15. Часы реального времени Ардуино DS1302 RTC / [Электронный ресурс] // РОБОТОТЕХНИКА18 : [сайт]. — URL: https://роботехника18.рф/ds1302-ардуино/ (дата обращения: 22.02.2024).

16. Arduino - Infrared Obstacle Avoidance Sensor / [Электронный ресурс] // Arduino Get Started : [сайт]. — URL: https://arduinogetstarted.com/tutorials/arduino-infrared-obstacle-avoidance-sensor (дата обращения: 26.04.2024).

17. Geddes M. Arduino Project Handbook, volume 2 [Текст] / Geddes M. — 1-е изд.. — Сан-Франциско: No Starch Press, 2017 — 326 с.

18. How to simply use DS1302 RTC module with Arduino board and LCD screen / [Электронный ресурс] // SurtrTech : [сайт]. — URL: https://surtrtech.com/2018/01/27/how-to-simply-use-ds1302-rtc-module-with-arduino-board-and-lcd-screen/ (дата обращения: 17.04.2024).

19. McGarry S. 3D Printing Basics: A Beginner’s Guide / McGarry S. [Электронный ресурс] // AutoDesk : [сайт]. — URL: https://www.autodesk.com/products/fusion-360/blog/beginners-guide-3d-printing-basics/ (дата обращения: 05.05.2024).

20. Monk S. Programming Arduino: Getting Started with Sketches (Tab) [Текст] / Monk S. — 2-е изд.. — Нью-Йорк: McGraw-Hill Education, 2016 — 208 с.

Приложение А

Код программы

```
//-------НАСТРОЙКИ---------
#define coin_amount 4    // число номиналов монет, которые нужно распознать
#define coin_amount_calibration 3  //  число монет одного номинала используемых при калибровке
float coin_value[coin_amount] = { 5.0, 2.0, 10.0, 1.0 };  // стоимость, номинал монет в порядке уменьшения диаметра!!!
String currency = "RUB"; // валюта (английские буквы!) BYN - Белорусские рубли
int displ_coin_value_time = 1500; // время отображения на дисплее номинала пролетевшей монеты (миллисекунды)
int stb_time = 2000;      // время бездействия, через которое система уйдёт в сон (миллисекунды)
//char* print_coin_value[coin_amount] = {"5 pyblei", "2 pybla", "10 ryblei", "1 pybl"}; // текстовый массив номиналов монет в порядке уменьшения диаметра!(английские буквы!)
char* print_coin_value[coin_amount]={"RU", "RU", "RU", "RU"}; // для вывода номиналов монет русскими буквами, смотри далее метку Dunay 5, пропиши номиналы монет

//-----ОБЪЯВЛЯЕМ ПЕРЕМЕННЫЕ И ИХ ТИП ДАННЫХ
int coin_signal[coin_amount];    // тут хранится максимальное значение сигнала пролетевшей монеты для каждого размера (номинала) монет
int coin_signal_min[coin_amount];    // тут хранится нижнее значение границы диаппазона сигнала для каждого размера (номинала) монет
int coin_signal_max[coin_amount];    // тут хранится верхнее значение границы диаппазона сигнала для каждого размера (номинала) монет
int coin_quantity[coin_amount];  // храним количество монет каждого номинала
int sens_signal, last_sens_signal; //значение уровня сигнала монеты в динамике от фотодатчика analogRead(IRsens)= analogRead(14)
byte empty_signal;              // храним уровень пустого/фонового сигнала
unsigned long standby_timer; //таймер ожидания (ухода в сон)
unsigned long reset_timer; //таймер очистки памяти, он так же используется для принудительного выхода из калибровки
unsigned long disvalcoin_timer; // таймер, времени вывода на дисплей номинала брошенной монеты.
float summ_money = 0;           // сумма монет в копилке
boolean recogn_flag, sleep_flag = true;   // флажки recogn_flag, sleep_flag = true (true (истина) значение отличное от 0)
boolean coin_flag = false; // coin_flag = false (ЛОЖЬ т.е 0)

//------ЗАГРУЖАЕМЫЕ БИБЛИОТЕКИ---------
#include "LowPower.h"
#include "EEPROMex.h"
#include "LCD_1602_RUS.h"
#include <DS1302.h>
LCD_1602_RUS lcd(0x27, 16, 2); // создать дисплей
DS1302 rtc(8, 7, 6);  //инициализация платы DS1302

//-------ПИНЫ---------
#define button 2         // кнопка "проснуться":D2, значениев программе =2
#define calibr_button 3  // скрытая кнопка калибровки и сброса: D3: значениев программе =3
#define piezo 5          // пищалка на D5
```

```
#define disp_power 12    // питание дисплея: D12, значениев программе disp_power=12
#define LEDpin 11        // питание ИК диода: D11, значениев программе LEDpin=11
#define IRpin 17         // питание фотодиода: A3, значениев программе IRpin=17
#define IRsens 14        // сигнал фотодиода: A0, значениев программе IRsens=14

void setup() {
  Serial.begin(9600);  // открыть порт для связи с ПК для отладки и задаем скорость работы монитора порта
  delay(500); // останавливаем выполнение программы на пол секунды
  tone(piezo,1000,500);
   void(* resetFunc) (void) = 0;//объявляем функцию reset с адресом 0 (для программного ресет)

  rtc.halt(false);
  rtc.writeProtect(false);

  // Если в DS1302 уже сохранено установленное время, то нужно за комментировать
 /* rtc.setDOW(MONDAY);          // Устанавливаем день недели MONDAY
  rtc.setTime(16, 53, 10);     // Устанавливаем время 20:30:10 (24часа формат времени)
  rtc.setDate(17, 06, 2024);   // Устанавливаем число месяц и год
 */

  // подтягиваем кнопки
  pinMode(button, INPUT_PULLUP);
  pinMode(calibr_button, INPUT_PULLUP);
  // пины питания как выходы
  pinMode(disp_power, OUTPUT);// питание дисплея
  pinMode(LEDpin, OUTPUT); //  питание ИК диода
  pinMode(IRpin, OUTPUT); // питание фотодиода
  //pinMode(A2, OUTPUT); //  резерв пин 16(A2) питание ещё чего либо
  // подать питание на дисплей и датчик
  digitalWrite(disp_power, 1); // или digitalWrite(disp_power, HIGH); питание дисплея
  digitalWrite(LEDpin, 1);   //  digitalWrite(LEDpin, HIGH); питание ИК диода пин D11,
  digitalWrite(IRpin, 1);   //  digitalWrite(IRpin, HIGH); питание фотодиода
    empty_signal = analogRead(IRsens);  // считать пустой (опорный/фоновый) сигнал empty_signal
  Serial.println("1.Void setup.Background signal level: empty_signal = " + String(empty_signal)); // 1-й раз вывод уровня фонового сигнала
  // инициализация дисплея
  lcd.init();
  lcd.backlight(); //  Включаем подсветку LCD дисплея
  if (!digitalRead(calibr_button)) {  // если при запуске нажата кнопка КАЛИБРОВКА
    for (byte i = 0; i < coin_amount; i++) {
        coin_quantity[i] = EEPROM.readInt(i * 2); // считываем количество монет каждого номинала, если они были сохранены ранее
    }
    // Вывод на дисплей "Сервис" и информации о процессе калибровки
    lcd.clear();
    lcd.setCursor(5, 0);
    lcd.print(L"Сервис");
    delay(500);
```

```
  reset_timer = millis(); //сбросить таймер
  while (1) {                          // бесконечный цикл
   if (millis() - reset_timer > 3000) {// если кнопка всё ещё удерживается и прошло 3 секунд,
     // то очистить количество монет в памяти
     for (byte i = 0; i < coin_amount; i++) {
         coin_quantity[i] = 0; // Память ОБНУЛЕНА! Количество монет в копилке всех номиналов - ноль
       EEPROM.writeInt(i * 2, 0);  // заполняем все ячейки памяти с адреса 0 по .... нулями
     }
     lcd.clear();
     lcd.setCursor(1, 0);
     lcd.print(L"Память очищена");
     delay(200);
   }
   if (digitalRead(calibr_button)) {   // если отпустили кнопку, перейти к калибровке
     lcd.clear();
     lcd.setCursor(3, 0);
     lcd.print(L"Калибровка");
     delay(1500);
     reset_timer = millis(); // сбросить таймер
     break;
   }
  }
  while (1) {
    for (byte i = 0; i < coin_amount; i++) { // последовательный перебор номиналов монет размер которых калибруется
      for (byte k = 0; k < coin_amount_calibration; k++) { // повторение калибровки монеты того же номинала (k раз)
       // вывод на дисплей
       lcd.setCursor(0, 1); lcd.print(coin_value[i]);  // отобразить цену монеты, размер которой калибруется
       lcd.setCursor(6, 1); lcd.print(currency);      // отобразить валюту
        lcd.setCursor(13, 1); lcd.print(coin_amount_calibration - k); // отобразить оставшееся количество попыток калибровки монеты того же номинала
        empty_signal = analogRead(IRsens);  // считать пустой (опорный/фоновый) сигнал empty_signal
            Serial.println("2.Calibration.Background signal level: empty_signal = " + String(empty_signal)); // 2-й раз вывод уровня фонового сигнала
          last_sens_signal = empty_signal;  // значение last_sens_signal = пустому (опорному/фоновому) сигналу empty_signal
       while (1) {  // бесконечный цикл сканирования фотодатчика и анализа полученного значения
         // если ничего не делали 60 секунд, то система выйдет из цикла калибровки программной перезагрузкой
        if (millis() - reset_timer > 60000) resetFunc(); //вызываем функцию reset

        sens_signal = analogRead(IRsens);     // считать фотодатчик
          if (sens_signal > last_sens_signal) last_sens_signal = sens_signal;  // если текущее значение больше предыдущего то запоминаем его
         if (sens_signal - empty_signal > 40) coin_flag = true;  // если, значение превысило фоновое на 5, считать что пролетает монета, ИСТИНА
```

```
          if (coin_flag && (abs(sens_signal - empty_signal)) < 10) { // если, значение упало, вернулось до фонового, то монета точно пролетела
              coin_signal[i] = last_sens_signal; // зафиксировали максимальное значение пролетевшей монеты
          if (k < 1) {
            coin_signal_max[i] = coin_signal[i];  // если эта была первая попытка, то ...
            coin_signal_min[i] = coin_signal[i];
          }
              if (coin_signal[i] > coin_signal_max[i]) coin_signal_max[i] = coin_signal[i]; // фиксируем максимальное значение
         if (coin_signal[i] < coin_signal_min[i]) coin_signal_min[i] = coin_signal[i]; // фиксируем минимальное значение
          coin_quantity[i]++;  // добавляем к общему количеству монет данного номинала ещё одну
          coin_flag = false;  // ЛОЖЬ
           reset_timer = millis(); //сбросить таймер программного reset для принудительного выхода из калибровки
          break; // принудительный выход из цикла while (1)
            } // конец, пролетающая монета отсканирована, возврат в непрерывный цикл сканирования фотодатчика, ждём пролёт следуещей монеты
        } // скобка цикла while (1) { } непрерывный цикл сканирования фотодатчика
         } // скобка цикла for (byte k = 0; k < coin_amount_calibration; k++), повторение калибровки монеты того же номинала (k раз)
     }// скобка цикла for (byte i = 0; i < coin_amount; i++), перебора номиналов монет размер которых калибруется
     break;// принудительный выход из цикла калибровки while (1)
   } // скобка while (1) { } цикла калибровки
   // Далее, исходя из полученных в цикле калибровки данных идёт расчёт значений верхних и нижних
   // границ диапазонов для каждого номинала монет, расширенных до их смыкания
    coin_signal_max[0] = 1023; // верхняя граница монеты с максимальным диаметром ограничена цифрой 1023
    coin_signal_min[coin_amount - 1] = empty_signal + 50 ; // нижняя граница монеты с минимальным диаметром на 15 единиц выше фонового значения
   for (byte i = 0; i < coin_amount; i++) {
    if (i > 0) {
     coin_signal_max[i] = (coin_signal_min[i - 1] - 1); //Верхняя граница диапазона номинала монеты
    }
    if (i < coin_amount - 1) { // пока переменная last_sens_signal свободна, используем её для вычеслений границ диапазонов!
     last_sens_signal = (coin_signal_min[i] - coin_signal_max[i + 1]) / 2; // величина увеличения границы диапазона, округленая до целого значения
      coin_signal_min[i] = coin_signal_min[i] - last_sens_signal; // нижняя граница диапазона номинала монеты
    }
   }
    // Записываем как массив данных расчитанные, расширенные границы диаппазонов распознования монет в память и количество монет
   for (byte i = 0; i < coin_amount; i++) {
    EEPROM.updateInt(i * 2, coin_quantity[i]); // обновляет байт данных в ячейке за
```

```
количество монет, если её старое содержимое отличается от нового
        EEPROM.writeInt(20 + i * 2, coin_signal_min[i]); // последовательно записали минимальное значение за каждую монету в ПАМЯТЬ в ячейку с 40 по ...
        EEPROM.writeInt(40 + i * 2, coin_signal_max[i]); // последовательно записали максимальное значение за каждую монету в ПАМЯТЬ в ячейку с 60 по ...
    }
    // вывод результата калибровки на дисплей
     for (byte i = 0; i < coin_amount; i++) { // отобразить на дисплее границы диапазонов распознавания для каждого номинала монет (но эти данные не из памяти EEPROM!)
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(coin_value[i]); lcd.print(L" "); // номинал монеты
         lcd.print(coin_signal_max[i]); lcd.print(L" -> "); lcd.print(coin_signal_min[i]); // и её диапазон
      lcd.setCursor(0, 1);
      if (i < coin_amount - 1) {
        lcd.print(coin_value[i + 1]); lcd.print(L" "); // номинал монеты
        lcd.print(coin_signal_max[i + 1]); lcd.print(L" -> "); lcd.print(coin_signal_min[i + 1]); // и её диапазон
       }
         delay(3000); // с паузой в 3 секунды, последовательно отображаются результаты калибровки за каждую монету
    }
    //---------------Конец вывода результатов калибровки-------------------
  }
  // Продолжается старт системы после калибровки (или минуя калибровку)
  // при старте системы считываем из памяти количество монет каждого номинала в копилке
  // а так же считать из памяти верхнее и нижнее значение диаппазона сигнала для каждого номинала монет
  for (byte i = 0; i < coin_amount; i++) {
      coin_quantity[i] = EEPROM.readInt(i * 2); // считывает количество монет каждого номинала
    coin_signal_min[i] = EEPROM.readInt(20 + i * 2); // считывает значение нижней границы диапазона
    coin_signal_max[i] = EEPROM.readInt(40 + i * 2); // считывает зачение верхней границы диапазона
       summ_money += coin_quantity[i] * coin_value[i]; // считаем сумму монет, как произведение количества монет на их номинал
  }
   Serial.println("Expanded coin recognition boundaries read from memory"); // Расширенные границы распознавания монет, считанные из памяти
  for (byte i = 0; i < coin_amount; i++) {
    Serial.println("coin_signal_max~min[" + String(i) + "]: " + String(coin_signal_max[i]) + "~" + String(coin_signal_min[i]));
  }
  Serial.println("3.Void setup.Background signal level: empty_signal = " + String(empty_signal));
  standby_timer = millis();  // обнулить таймер ухода в сон
} // Конец выполнения функции void setup
void loop() {
  if (sleep_flag) {
 lcd.clear();
```

```
    lcd.setCursor(0, 0);
    lcd.print(rtc.getDateStr());
    lcd.setCursor(13, 0);
    lcd.print(currency);
    lcd.setCursor(0, 1);
    lcd.print(rtc.getTimeStr());
    lcd.setCursor(11, 1);
    lcd.print(summ_money);
   delay(1000);
    empty_signal = analogRead(IRsens); // считать датчик (считать пустой (опорный/фоновый) сигнал empty_signal)
    Serial.println("4.Void loop.Background signal level: empty_signal = " + String(empty_signal)); // 4-й вывод уровня фонового сигнала
    sleep_flag = false;
  }
  last_sens_signal = empty_signal;  // максимальное значение сигнала последней пролетевшей монетки last_sens_signal сбрасывается до уровня фона empty_signal
  // Далее работаем в бесконечном цикле
   // While будет работать в цикле непрерывного сканирования до тех пор, пока время таймера ухода в сон не вышло и break
  while (1) {
    // далее такой же алгоритм, как при калибровке
    sens_signal = analogRead(IRsens);  // считать датчик
    if (sens_signal > last_sens_signal) last_sens_signal = sens_signal;  // если текущее значение больше предыдущего
     if (sens_signal - empty_signal > 5) coin_flag = true;  // если значение упало почти до "пустого", считать что монета улетела
    if (coin_flag && (abs(sens_signal - empty_signal)) < 4) {    // если монета точно улетела
      recogn_flag = false;  // флажок ошибки, пока что не используется
      for (byte i = 0; i < coin_amount; i++) {
        if ( last_sens_signal >= coin_signal_min[i] && last_sens_signal <= coin_signal_max[i]) {
          summ_money += coin_value[i];
          coin_quantity[i]++;  // для распознанного номера монетки прибавляем количество
          lcd.clear();
            lcd.setCursor(4, 0); lcd.print(print_coin_value[i]); // Номинал распознанной монеты прописью
            // lcd.setCursor(0, 0); lcd.print(coin_value[i], 2);  // Номинал распознанной монеты цифрой
          lcd.setCursor(0, 1); lcd.print(summ_money);//  вывести сумму монет
          lcd.setCursor(13, 1); lcd.print(currency);// отобразить валюту
            //lcd.setCursor(12, 1); lcd.print(last_sens_signal);lcd.print(L"  "); // уровень сигнала пролетевшей монеты (значения от фона до 1023)
          if (i == 0 && print_coin_value[i] == "RU") {
            tone(piezo,1500,700);
            lcd.setCursor(0, 0);
            lcd.print(L"    5 рублей     ");
          }
          if (i == 1 && print_coin_value[i] == "RU") {
            tone(piezo,500,500);
            lcd.setCursor(0, 0);
            lcd.print(L"   2 рубля    ");
```

```
        }
        if (i == 2 && print_coin_value[i] == "RU") {
          tone(piezo,2000,1500);
          lcd.setCursor(0, 0);
          lcd.print(L"    10 рублей     ");
        }
        if (i == 3 && print_coin_value[i] == "RU") {
          tone(piezo,1000,200);
          lcd.setCursor(0, 0);
          lcd.print(L"   1 рубль    ");
        }
        recogn_flag = true;  // ИСТИНА значение отличное от 0
         break;// Break используется для принудительного выхода из цикла, for (byte i = 0; i < coin_amount; i++). Монета опознана, не надо перебирать другие границы.
       }  //  Монета попала в диапазон, распознана, посчитана и выведена на дисплей. Break выходим из цикла for (byte i = 0; i < coin_amount; i++)
     } // конец цикла for (byte i = 0; i < coin_amount; i++) перебора и сравнения сигнала пролетевшей монетой с хранящимися в памяти
    coin_flag = false;
    standby_timer = millis();  // сбросить таймер,  millis() количество миллисекунд с момента начала выполнения программы
    break;
   }
   if (millis() - standby_timer > stb_time) {
    // good_night();
    sleep_flag = true;
    break;
   }
    while (!digitalRead(button)) { // нажата кнопка "проснуться" (или монетка замыкает контакты)
    if (millis() - standby_timer > 3000) { // если контакты "проснуться" замкнуты > 3 секунд, то вывод статистики
      lcd.clear();
      for (byte i = 0; i < 4; i++) { // отобразить на дисплее статистику для первых 4-х монет
       lcd.setCursor(i * 4, 0); lcd.print(coin_value[i], 1); // сверху номенал монеты (округлен до десятых)
       lcd.setCursor(i * 4, 1); lcd.print(coin_quantity[i]); // снизу их количество

      }
      delay(3000);
    } // скобка if (millis() - standby_timer > 5000){ когда если контакты "проснуться" замкнуты > 5 секунд, то вывод статистики }
   } // скобка while (!digitalRead(button)) { когда нажата кнопка "проснуться"  }
  } // скобка while (1) { бесконечного цикла сканирования датчиков }
} //скобка цикла void loop() {  }
```

Приложение Б

Действующий макет электронного счетчика монет.

Студент:	Руководитель:	ТГУ 2024
Катин С.А.	Прядилов А.В.	

Инструкция по демонстрации стенда

1. Установить стенд на столе.

2. На задней стороне стенда переключить рычажок On/Off в верхней правой части корпуса. Если дисплей не загорелся подключить стенд шнуром micro-USB к источнику питания.

3. Вывод времени:

- SW5 (настройка): отвечает за переключение режимов настройки времени/вывода времени на экран.

- SW1/SW4 (вывод времени/даты): при нажатии на SW1 выводится настроенные часы, при нажатии на SW4 выводится настроенная дата

- SW2/SW3 (увеличение/уменьшение параметра): при нажатии на SW2 настраиваемый параметр увеличивается, при нажатии на SW3 настраиваемый параметр уменьшается

4. Режим считывания монет включается поднесением монеты в отверстие монетоприемника и замыканием монетой пластин в отверстие. После чего в отверстие можно кидать монетки для считывания.

Описание стенда

Электронный счетчик монет предназначен для демонстрации работы схемы с монетоприемником и часами. Стенд выполнен на основе микроконтроллера Arduino Nano. Он является одновременно и копилкой, и часами. Такое устройство может найти применение как дома, так и на работе в офисе. Везде где у людей есть немного места на столе для размещение данного устройства.