Проектирование устройств

Домашнее задания №1

Евгений Зеленин

18 ноября 2024 г.

1. ДЗ №1 Знакомство с Arduino

Условия задачи. *Необходимо подключить механическую кнопку и светодиод. Напишите программу без каких либо средств устранения дребезга, посмотрите что получится. Затем попробуйте различные варианты подтяжки(программный вариант, аппаратный вариант - о них мы говорили на занятии) или же свой вариант с опросом кнопки через короткие промежутки времени(детально этот вариант мы не прорабатывали, поэтому это по желанию).*

Подключение энкодера

Выполнение этого дз решил начать с подключения энкодера, чтобы закрепить материал из лекции. Схема подключения взята из лекции, на рисунке можно увидеть подключение на Breadboard.

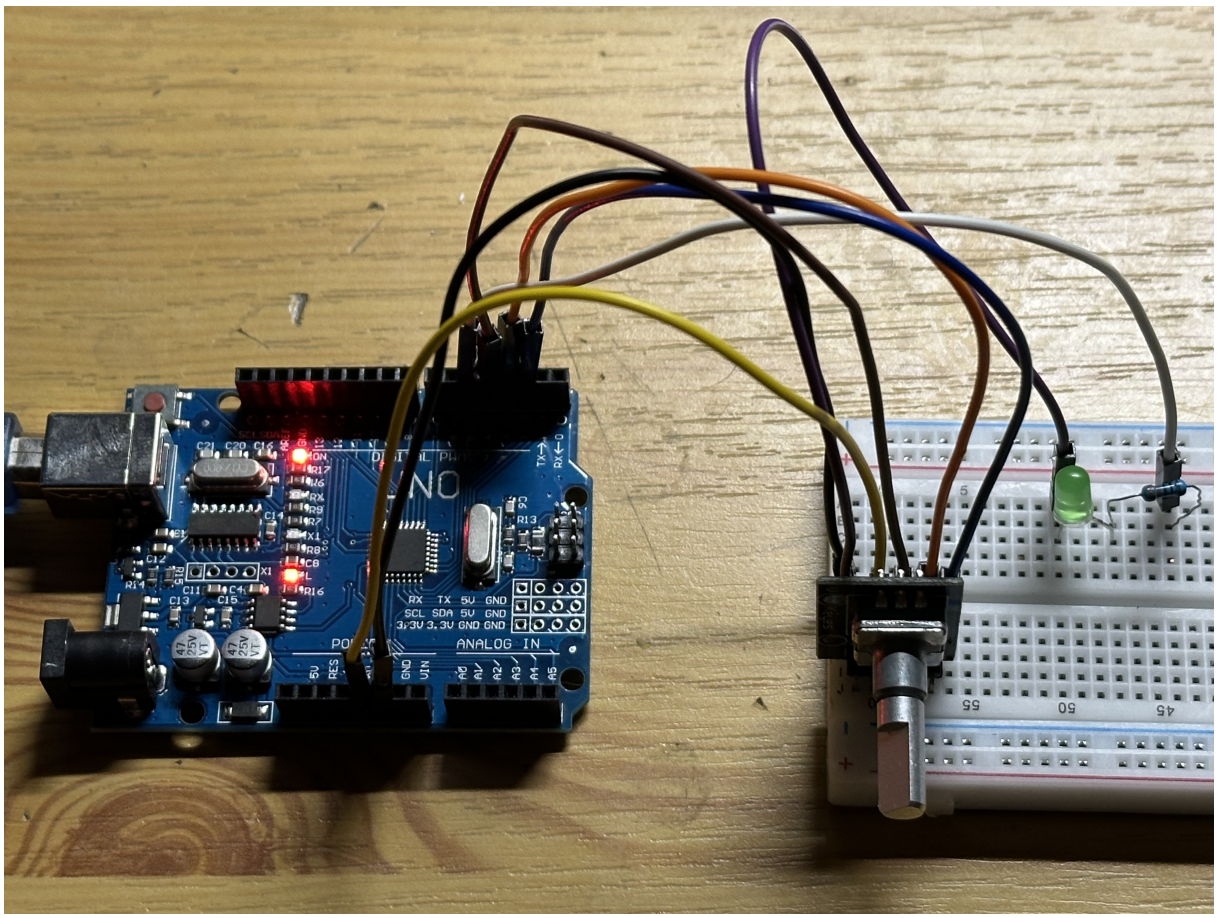

Рис. 1: Подключение энкодера на breadboard

Чтобы проверить как работает энкодер, использовал код из лекции. Все работает замечательно, пришлось лишь добавить INPUT_PULLUP для корректной работы кнопки.

```
#include "RotaryEncoder.h"          // библиотека для энкодера
int buttonState = 0;
uint8_t pin = 2;
int cnt = 0;
static int pos = 0;  // задаем начальное положение энкодера
int newPos;
RotaryEncoder encoder(2, 3, RotaryEncoder::LatchMode::TWO03);  // пины подключение энкод
void setup() {
  pinMode(4, INPUT_PULLUP); // режим работы цифрового вывода для кнопки
  pinMode(5, OUTPUT);
  Serial.begin(9600);
  Serial.println(pos);  // выводим на монитор начальное значение
}

void loop() {
   // проверяем положение ручки энкодера
   encoder.tick();
   newPos = encoder.getPosition();

   // если положение изменилось - выводим на монитор
   if (pos != newPos) {
      Serial.println(newPos);
      pos = newPos;
   }
   if (!digitalRead(4) == HIGH) {
    digitalWrite(5, HIGH);
   } else {
    digitalWrite(5, LOW);
   }
}
```

Видео с демонстрацией работы энкодера приложено к занятию и находится в папке с материалами на Google Disk, ссылка в конце отчета.

Подключение кнопки

Подключение кнопки сделано в соответствии со схемой из лекции, на рисунке 2 - реализация на breadboard.

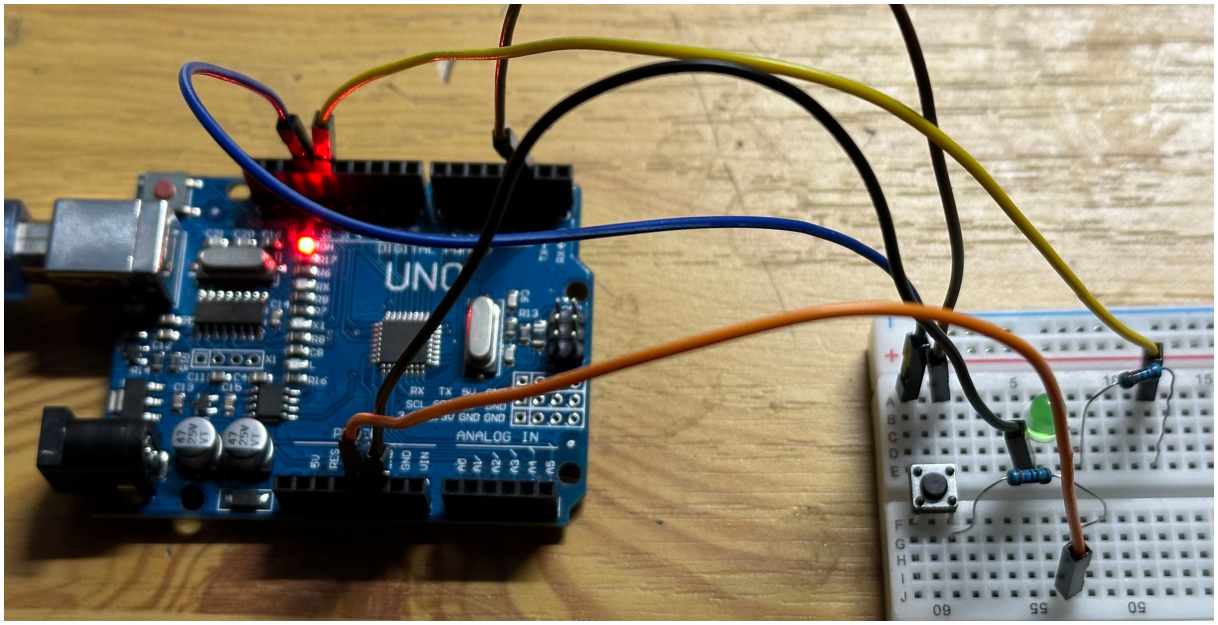

Рис. 2: Подключение кнопки на breadboard

Сначала, попробовал работу кнопки без подтяжки к +5в. Кнопка не работает. После этого, сделал вывод в терминал через COM-порт, в котором было видно хаотичные срабатывания. После включения встроенной подтяжки через INPUT_PULLUP кнопка заработала как надо.

Также, был проведен эксперимент с использованием внешнего подтягивающего резистора на 10кОм, разницы в работе со встроенным резистором не замечено.

В код, в самом конце, добавил задержку 1мс, для того чтобы посчитать длительность нажатия и вывести ее в терминал.

```
int buttonState = 0;
uint8_t pin = 2;
int cnt = 0;

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(12, OUTPUT);
}


void loop() {
 buttonState = digitalRead(pin);
```

```
 if (buttonState == LOW) {
    digitalWrite(13, HIGH);
    ++cnt;
 }
 else {
  if(cnt){
     Serial.print("\n");
     Serial.print(cnt);
  }
  digitalWrite(13, LOW);
  cnt = 0;
 }
  delay(1);
}
```

Далее, провел такой эксперимент. Подключил осциллограф и измерил длительность нажатия кнопки, сравнил с тем, что получилось на Ардуино.

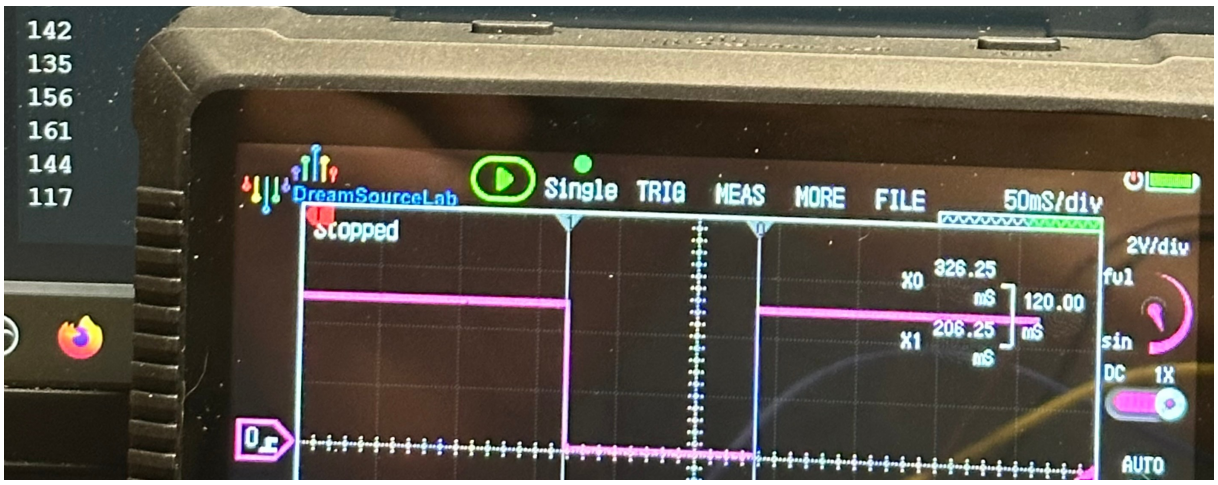

Рис. 3: Подсчет длительности нажатия кнопки

Длительность нажатия кнопки, измеренная осциллографом, в среднем, от 100 до 200мс. Разница между подсчетами Ардуино и осциллографом составили 3мс (при 120мс на осциллографе, в терминал выводится значение 117мс). Разницу в 1мс можно объяснить дискретностью подсчетов в 1мс, еще 1мс можно списать на погрешность измерения, 1мс на задержки при работе с портами ввода-вывода и COM-портом.

Также, была осуществлена проверка алгоритма фильтрации нажатий следующим методом:

```
  enum { button_read_period = 10 }; /* опрашивать состояние будем каждые 10 мс
  void loop()
  {
    static uint8_t  button_state = 0; //* Состояние кнопки; каждый бит соответствует coc
    {
        static uint32_t ts = 0; //* здесь храним момент последнего отсчёта
```

```
        if ( millis() - ts > button_read_period ) //* Если с момента последнего отсчёта
        {
            ts = millis();   //* Запоминам этот момент
            button_state <<= 1; //* сдвигаем текущее состояние на один бит влево, «забыв
            button_state  |=  digitalRead(pin); //* И в крайний правый бит записываем те
        }
    }

    switch(button_state)
    {
        case  0xff:  //*то есть все биты «1» - последние 8 отсчётов кнопка была отпущена
            //*Кнопка отпущена в течение минимум (button_read_period * 8) миллисекунд
            Serial.print("RELEASED\n");
            digitalWrite(12, LOW);
            break;
        case 0:  // то есть все биты «0» - последние 8 отсчётов кнопка была нажата
              //*Кнопка нажата в течение минимум (button_read_period * 8) миллисекунд
            Serial.print("PRESSED\n");
            digitalWrite(12, HIGH);
            break;
        default:
            //*Кнопка в процессе переключения;
            Serial.print("SWITCHING\n");
            digitalWrite(12, HIGH);
            break;
    }
}
```

Алгоритм работает корректно, для наглядности был добавлен вывод в COM-порт.

Устранение дребезга контактов

Чтобы устранить дребезг контактов можно воспользоваться как программным вариантом, так и аппаратным. В качестве программного варианта можно использовать следующий код:

```
bool flag = false;
uint32_t btnTimer = 0;
void loop() {
  // читаем инвертированное значение для удобства
  bool btnState = !digitalRead(pin);
  if (btnState && !flag && millis() - btnTimer > 100) {
    flag = true;
    btnTimer = millis();
    Serial.println("press");
  }
  if (!btnState && flag && millis() - btnTimer > 100) {
    flag = false;
```

```
      btnTimer = millis();
      //Serial.println("release");
    }
  }
```

Алгоритм "отсекает"первые 100мс, пока нажата кнопка. Работает корректно, для наглядности был добавлен вывод в COM-порт.

Также, можно использовать и аппаратный вариант устранения дребезга контактов. Схема подключения кнопки в таком случае, выглядит следующим образом:

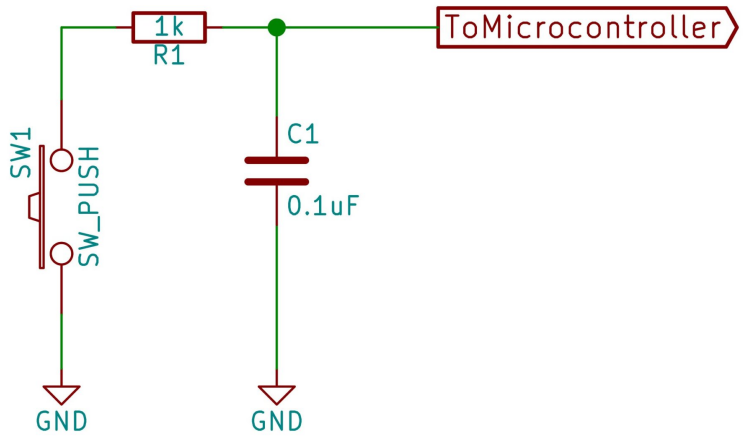

Рис. 4: Устранение дребезга контактов аппаратным способом

Этот способ также работает корректно, ложные срабатывания исчезают (рисунок 5).

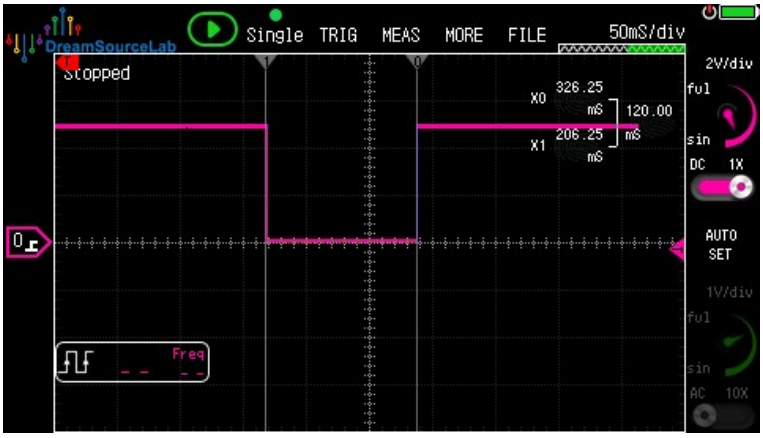

Рис. 5: Осциллограмма после установки RC-цепочки

2. Материалы к занятию

Схемы, демонстрация работы и прочие материалы к занятию расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №1