Портативное устройство вывода изображений на экран осциллографа.

Ильясов Александр, ФМЛ№30, 11-1

2017 г.

Содержание

1 Основная идея и ЦАП

1.1 Общие слова

Любой микроконтроллер способен выдавать только цифровой сигнал на свои пины. Одним из способов эмуляции аналогового выходного сигнала является использование ШИМ (PWM), порты с его поддержкой есть на плате arduino uno. Однако, частоты сигнала ШИМа не хватает, чтобы использовать его с осциллографом (и вряд ли вообще возможно сделать ШИМ такой частоты, чтобы он корректно работал с ним). Потому для поставленной задачи нужно использовать т.н. ЦАП (DAC) - цифро-аналоговый преобразователь - схему, которая принимает на вход цифровой сигнал и преобразовывает его в аналоговый. На arduino uno таких портов нет (есть только входные порты с АЦП (ADC), осуществляющим обратное преобразование), но некоторые виды ЦАПов можно изготовить самому. Я выбрал ЦАП на R-2R матрице (R-2R resistor ladder). Вот его схема:

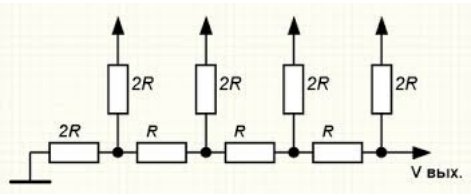


Рис. 1: R-2R матрица

Основные преимущества такого ЦАПа в том, что:

- Он прост в изготовлении и понимании его принципа работы
- Он осуществляет преобразования мгновенно
- Его битность может быть очень легко увеличена/уменьшена
- Он включает только дешевые и легкодоступные компоненты

1.2 Принцип работы R-2R матрицы

Для начала докажем, что схема имеет постоянный импеданс, равный R. Для этого применим преобразования Тевенена для последовательно для каждого из цифровых входов:

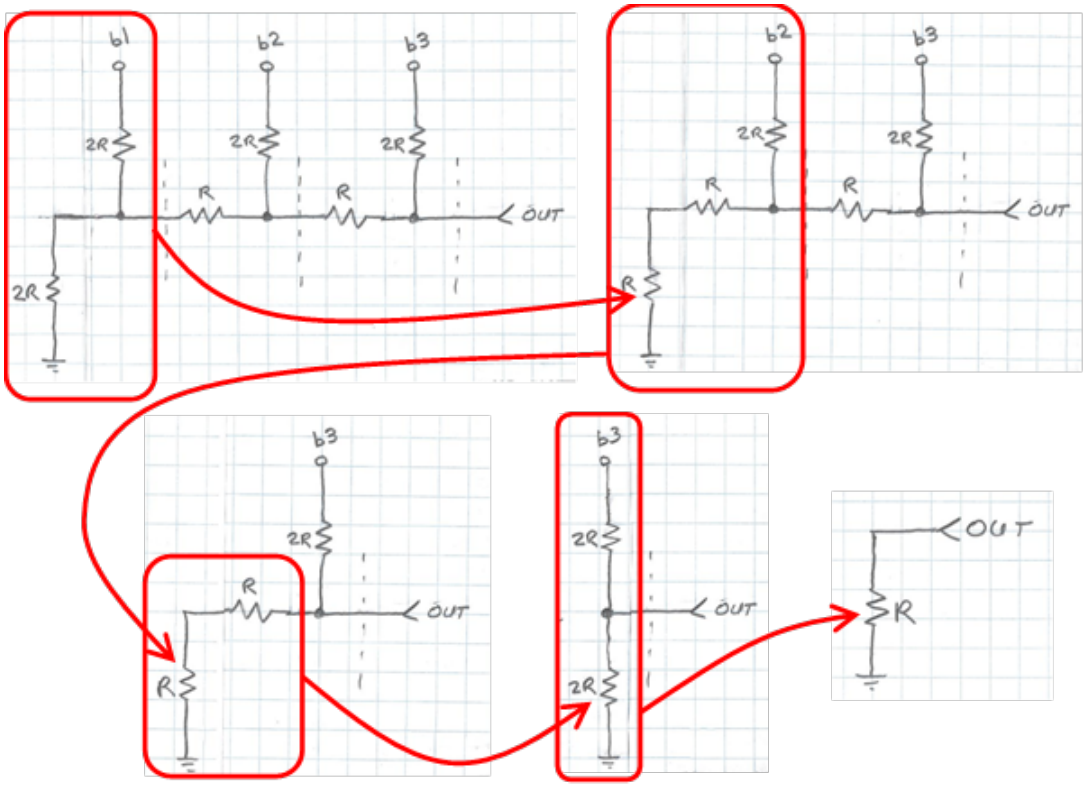


Рис. 2: Преобразования Тевенена, когда все порты в состоянии логического нуля

Мы убедились, что сопротивление такой схемы равно R. Стоит заметить, что сопротивление никак не именится, если некоторые порты перевести в состояние логической единицы: добавятся лишь источники ЭДС.

Рассмотрим такой случай. Допустим, в состояние логической единицы переведены порты b_0 и b_2 (напряжение логической единицы на arduino uno равно 5 V, потому далее напряжение в 5 V абсолютно эквивалентно по смыслу напряжению логической единицы).

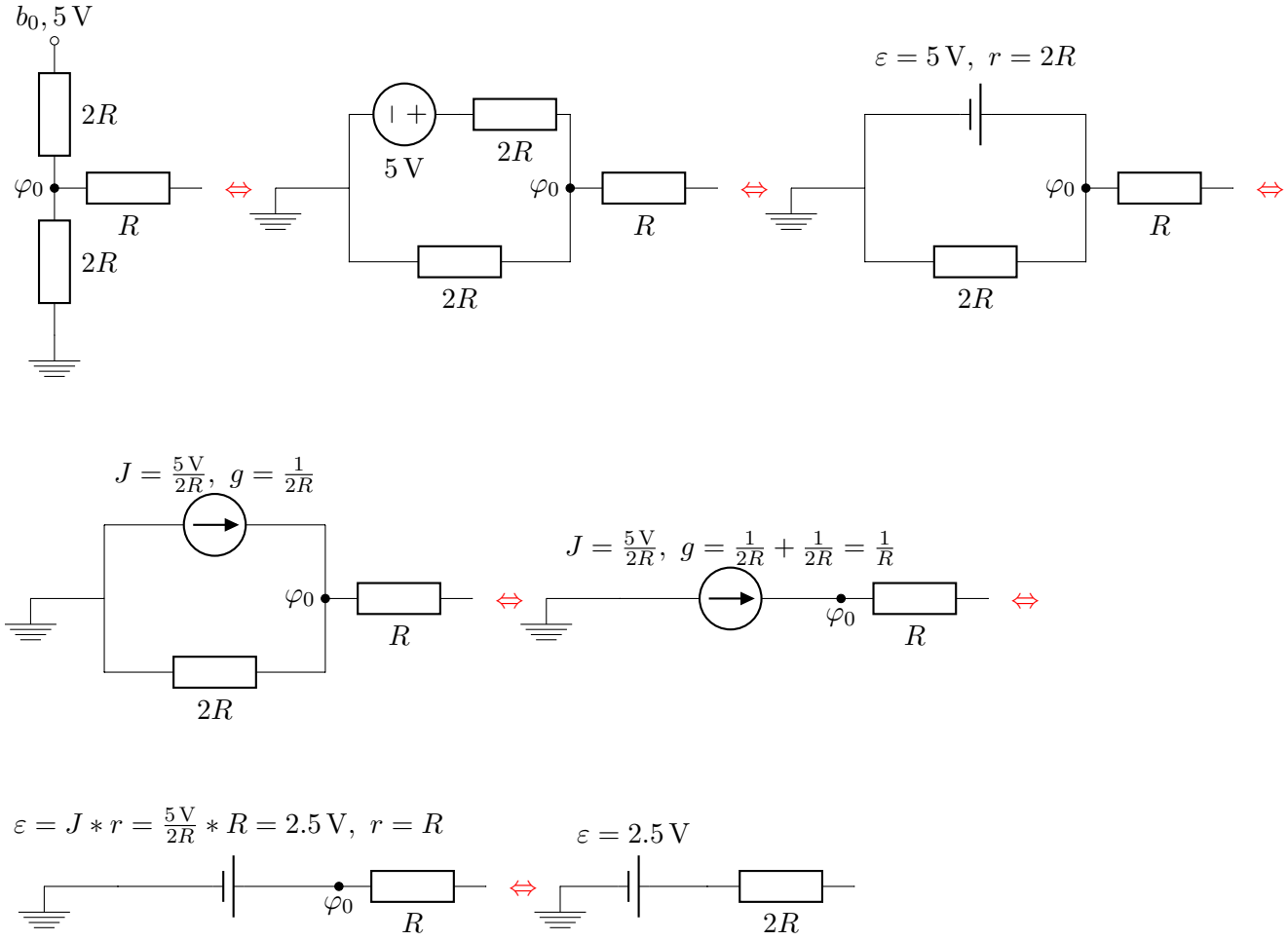


Рис. 3: Преобразования схемы для нулевого порта

Получили, что вся цепь левее конца резистора $2R$, подсоединённого к порту b_1, представима в виде идеального источника с ЭДС 2.5 V и последовательного резистора номиналом $2R$.

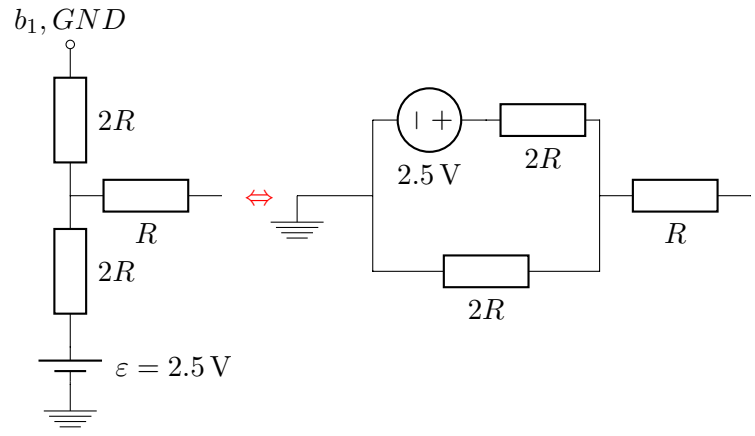


Рис. 4: Преобразования схемы для порта с логическим нулем

Нетрудно заметить, что получилась та же схема, которая получилось после первого шага в преобразовании для нулевого порта с той лишь разницей, что теперь ЭДС начального источника вдвое меньше, а потому в конце ЭДС эквивалентного источника также будет в два раза меньше и равна 1.25 V.

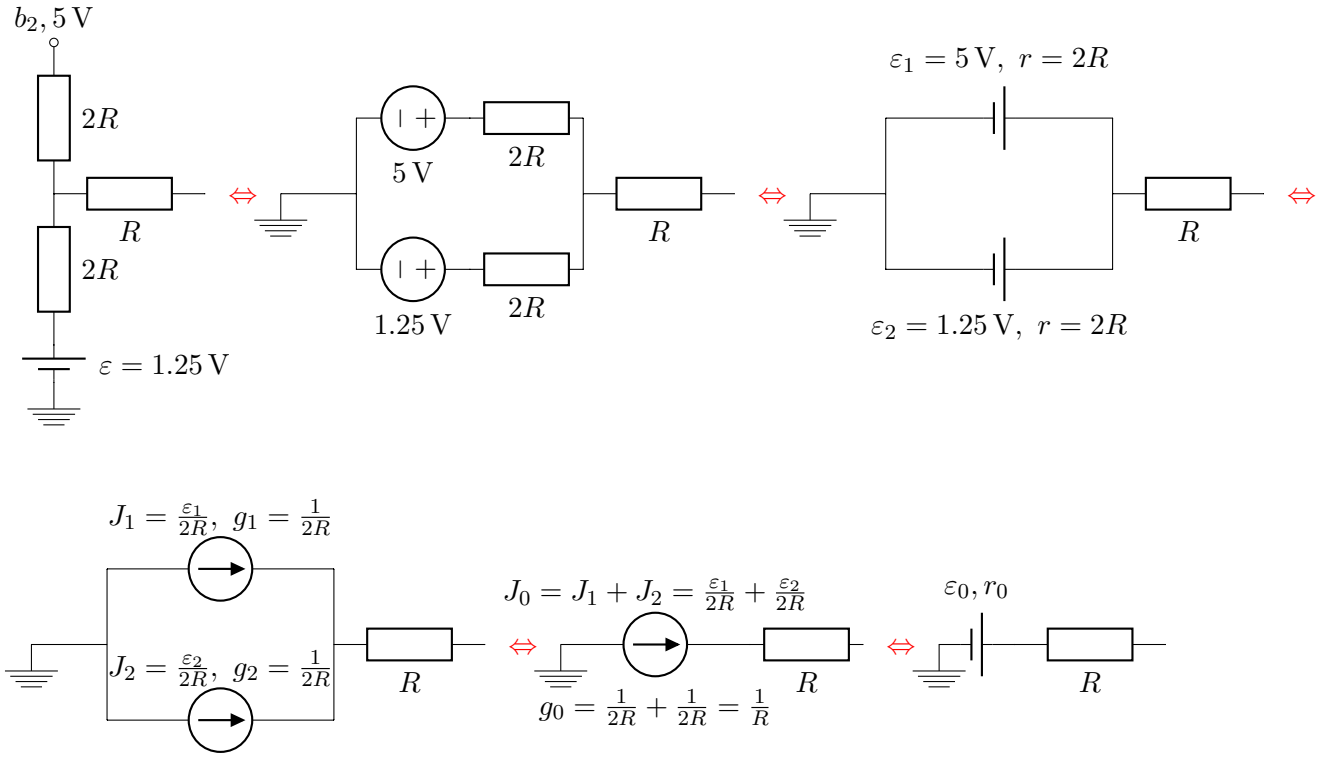


Рис. 5: Преобразование схемы для порта с логической единицей

$r_0 = \frac{1}{g_0} = R,\ \varepsilon_0 = J_0 * r_0 = \frac{\varepsilon_1+\varepsilon_2}{2R} * R = \frac{\varepsilon_1}{2} + \frac{\varepsilon_2}{2}$

Таким образом, видим, что при продвижении по схеме от нулевого порта с каждым следующим портом суммарное ЭДС эквивалентного источника увеличивается на $5V$ (если следующий порт в состоянии логической единицы) и потом уменьшается в два раза (понятно, что если следующий порт в состоянии логического нуля, то ЭДС просто делится пополам).

Иными словами, напряжение с нулевого порта делится пополам ровно столько раз, сколько всего входов в данном ЦАПе, с первого порта - на один раз меньше и т.д., напряжение с последнего порта делится только один раз.

Для n-разрядного ЦАП выходное напряжение равно:

$$U_{out} = \sum_{i=0}^{n} \frac{U_{b_i}}{2^{n-i}} = \sum_{i=0}^{n} (\frac{U_{b_i}}{2^n} * 2^i) = \sum_{i=0}^{n} (U_{min} * b_i * 2^i) = U_{min} * \sum_{i=0}^{n} (b_i * 2^i) = U_{min} * \overline{b_n b_{n-1} \dots b_1 b_0}$$

Получается, что выходное напряжение нашего ЦАПа пропорционально входному сигналу. Входной сигнал целый и лежит в пределах $[0;\ 2^n - 1]$ и для получения желаемого результата нужно подать на входные порты ЦАПа последовательность нулей и единиц, соответствующую его записи в двоичной системе счисления. Коэффициентом пропорциональности зависимости выходного напряжения от входного сигнала будет минимальное ненулевое выдаваемое напряжение ЦАПа. Оно равно $\frac{5\,\mathrm{V}}{2^n}$ и соответствует режиму работы, когда в состоянии логической единицы только нулевой порт.

1.3 Реализация ЦАП

1. В качестве платы управления была выбрана arduino uno по причине её доступности и простоты в использовании.

2. Чтобы выводить изображение необходимо два независимых ЦАПа - на y-канал и на x-канал.

3. Плата имеет 14 цифровых выходов, однако нулевой и первый порты используются при передаче данных через последовательный порт, потому их было решено не использовать. В результате осталось портов на два 6-разрядных ЦАПа.

4. Максимальный ток, втекающий в землю такого ЦАПа: $I_{max} = \frac{\frac{5\,\mathrm{V}}{64} * 63}{2R}$, при $R = 1\,\mathrm{k\Omega}$ он равен $\sim 2.5\,\mathrm{mA}$, а максимальный ток через порт arduino uno равен $20\,\mathrm{mA}$. Поэтому я решил использовать именно такое сопротивление.

5. Сначала был тестовый образец, собранный на макетке, к каждому порту был параллельно с основной цепью подсоединён светодиод для проверки правильности кода (порты переводятся в нужное состояние), измерение мультиметром разности потенциалов на выходе и на земле платы arduino позволило проверить работу ЦАПа.

6. Далее следовало моделирование в excel разводки ЦАПа на плате Perfboard и распайка в соответствии с ней.

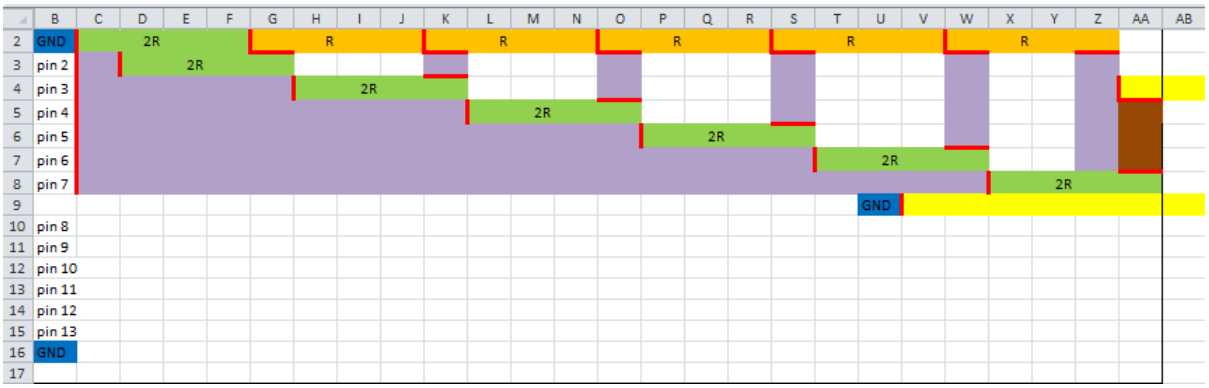
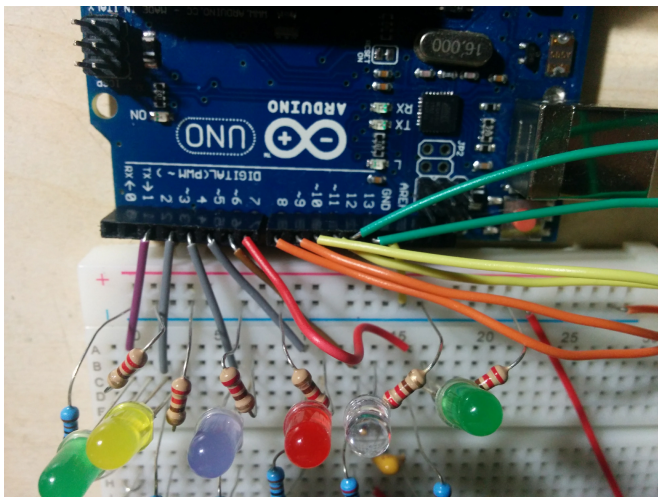


Рис. 6: Разводка ЦАПа в excel

7. Каждая ячейка - отверстие платы Perfboard, красные отрезки - места соединения соседних отверстий, фиолетовый - соединение проводником, жёлтый - выводы(на схеме один - на один из ЦАПов, второй для общей земли), коричневый - подключенный параллельно выводу фильтрующий конденсатор, в необходимости которого я не уверен. На схеме изображён только один ЦАП, второй получается зеркальным отражением первого. Технически в каждый пин вставлен припаянный к плате штырёк, земля на ЦАПы взята с порта, следующего за последним(13-ым) цифровым, к осциллографу - с колодок power.



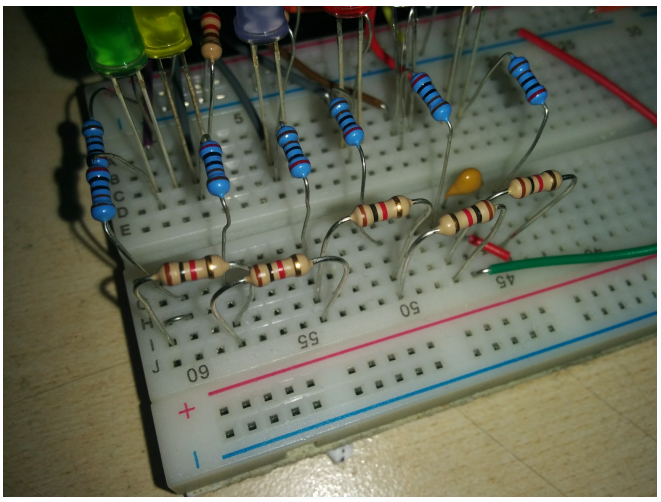


Рис. 7: Тестовый образец ЦАПа на макетке

2 Программная часть

2.1 Использованное ПО

Программа для arduino написана в arduino IDE, для взаимодействия компьютера с arduino (для тестов и загрузки изображений) - на Processing, поскольку он идуально совместим с arduino и позволяет легко работать с изображениями.

2.2 Логика программ

Для работы с платой планировалось использовать следующую систему: программа на arduino для приёма по USB и сохранения изображений, программа для компьютера для передачи изображения и ещё одна программа для arduino только для вывода изображения. Загружается картинка следующим образом: сначала на arduino нужно загрузить программу 4.2, после загрузки она ожидает начало перелачи данных, затем нужно запустить на компьютере, подсоединённом к плате по USB, программу 4.1, после чего начинается передача картинки. При этом в консоль Processing'a записывается отладочная информация о первых 100 пикселях; туда же выводится сообщение об окончании передачи. После этого можно загружать программу 4.3 на ардуино, предварительно остановив выполнение программы на Processing'е. Далее достаточно подсоединить плату к осциллографу и источнику питания и настроить осциллограф. Такая схема нужна для того, чтобы при уже загруженном изображении можно было вывести его просто подключив к плате источник питания(например, крону) и подсоединив её к осциллографу.

2.3 Этапы работы

1. Была написана программа для вывода произвольной точки по её координатам - была специальная функция, получавшая x и y координаты и в цикле пробегала по всем битам этих чисел (путём смещения числа вправо на <номер итерации> бит) и записывала в порт . В тот момент ЦАПы ещё были собраны на макетке и снабжены светодиодами, так что можно было проверить правильность подачи сигнала на порты, а подсоединение к осциллографу показало, что ЦАП работает так, как иожидалось, расстояние между соседними точками одинаковое вдоль обеих осей.

2. Следующей задачей было загрузить картинку на arduino. Для удобства загрузки и возможности хранить несколько картинок я решил использовать EEPROM - энергонезависимую память arduino uno, объём которой составляет 1024 B, что позволяет хранить до 2 картинок с максимальным разрешением ЦАПов - 64×64 (размер такой картинки равен $64 * 64\,\text{bit} = 8 * 64\,\text{B} = 512\,\text{B}$) либо до 8 разрешением 32×32 (размер - 128 B). Были написаны программы для приёма изображения платой arduino (см. код 2 - в разделе 4.2) и передачи его с компьютера (см. код 1 - в разделе 4.1).

3. После загрузки картинки она представляет из себя массив char'ов размером 8×64, в котором первые 8 элементов - первая строка, далее вторая и т.д.; при этом каждый элемент отвечает за 8 последовательных координат, а каждый бит в нём показывает, должна ли гореть точка с соответствующей координатой или нет. Например, нулевой бит нулевого элемента показывает состояние самой верхней левой точки, первый бит - первой слева; нулевой бит восьмого эемента - самой левой точки второго свреху ряда и т.д.

4. Далее была написана программа 3 - для вывода изображения. В ходе её выполнения сначала из EEPROM загружается картинка - массив char'ов. После этого в каждой итерации бесконечного цикла перебирается каждый элемент массива и в нём перебирается каждый его бит. Если бит равен единице, то на ЦАПы подаётся соответствующие x и y координаты, иначе - ничего не происходит. О том, как подаётся сигнал на пины - в п. 3 раздела 3.2.

3 Особенности работы и найденные фичи

3.1 Настройка осциллографа

Для работы платы необходим двухканальный осциллограф. Его нужно перевести в режим x-y и подсоединить плату как показано на картинке. Провод посередине - к земле. Он нужен, чтобы изображение не деформировалось из-за различных шумов. Оба канала выставляются на 0.5 V или 1 V на деление, в зависимости от желаемого размера изображения. Важно выключить растяжку и перевести оба канала в режим $\simeq$ - постоянное + переменное напряжение, поскольку быстрая его смена при работе ЦАПа приводит к тому, что по постоянному напряжению картинка впринципе не выводится, а по переменному - сильно искажается. Остальные ручки и переключатели могут служить либо для смещения изображения на экране, либо ни на что не влияют.

Рис. 8: Правильная настройка осциллографа и подсоединение платы

3.2 Встреченные баги и их решение

1. Большой проблемой была очень **долгая** (до 10 min) **компиляция** и загрузка скетча, притом проблема проявлялась только в школе. Вопреки логичному предположению, это был не дух физлаба, а антивирус. Дело в том, что при компиляции скетча Arduino IDE создаёт множество маленьких служебных файлов, проверка которых антивирусом может занимать много времени. Только в школе эта проблема проявлялась скорее всего потому, что школьная сеть была сохранена как общественная и это приводило к более активному поведению антивируса. **Решение** проблемы - добавление папки с Arduino IDE и самой программы в исключения антивируса; **также** можно в настройках Arduino IDE отключить опцию "Проверить код после вгрузки это устранит одну из проверок правильности закрузки кода, которых и без того достаточно (понятно, что если что-то пойдёт не так, её можно включить обратно).

2. Также стоит заметить, что если на компьютере запущена программа на Processing, то это делает невозможным загрузку скетча. В этом случае нужно сначала остановить программу на Processing'е, а уже потом вгружать скетч.

3. Одной из самых больших проблем было то, что в первой версии вывода изображения использовалась встроенная функция digitalWrite() (см. функцию pinPut() в коде 3). Недостаток этого был в том, что на картинке появлялись лишние точки, а частота кадров и **быстродействие** были очень **низкими**. Причина этого в том, что функция digitalWrite() на самом деле содержит больше десятка строк кода и выполняется большое для этой задачи время. **Решение** этой проблемы в записи в регистры процессора. Вкратце, у процессора есть т.н. регистры - память, в которой он хранит служебную информацию. В процессоре Arduino uno (ATmega 328) есть три регистра, отвечающих за взаимодействие с пинами: B, C, D.

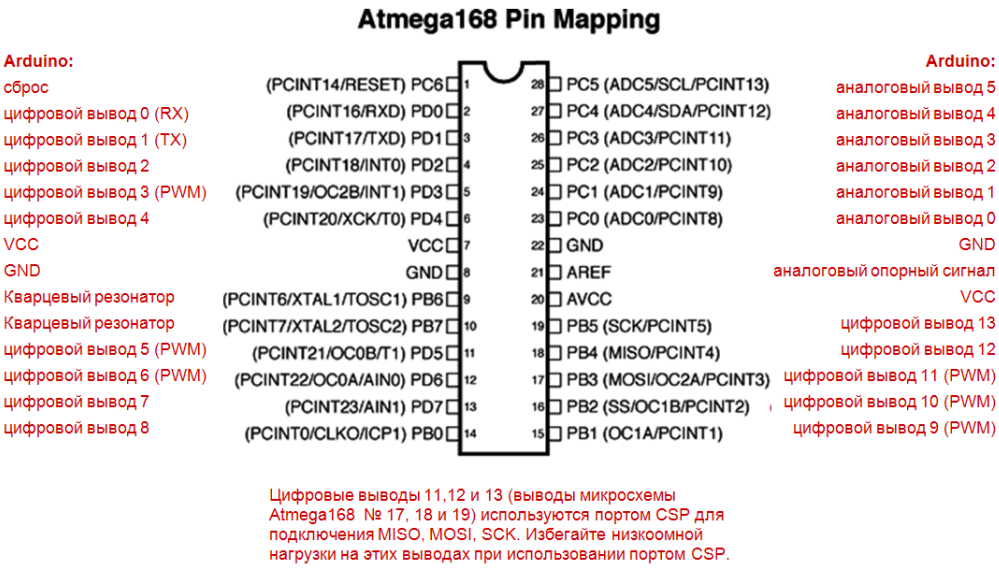
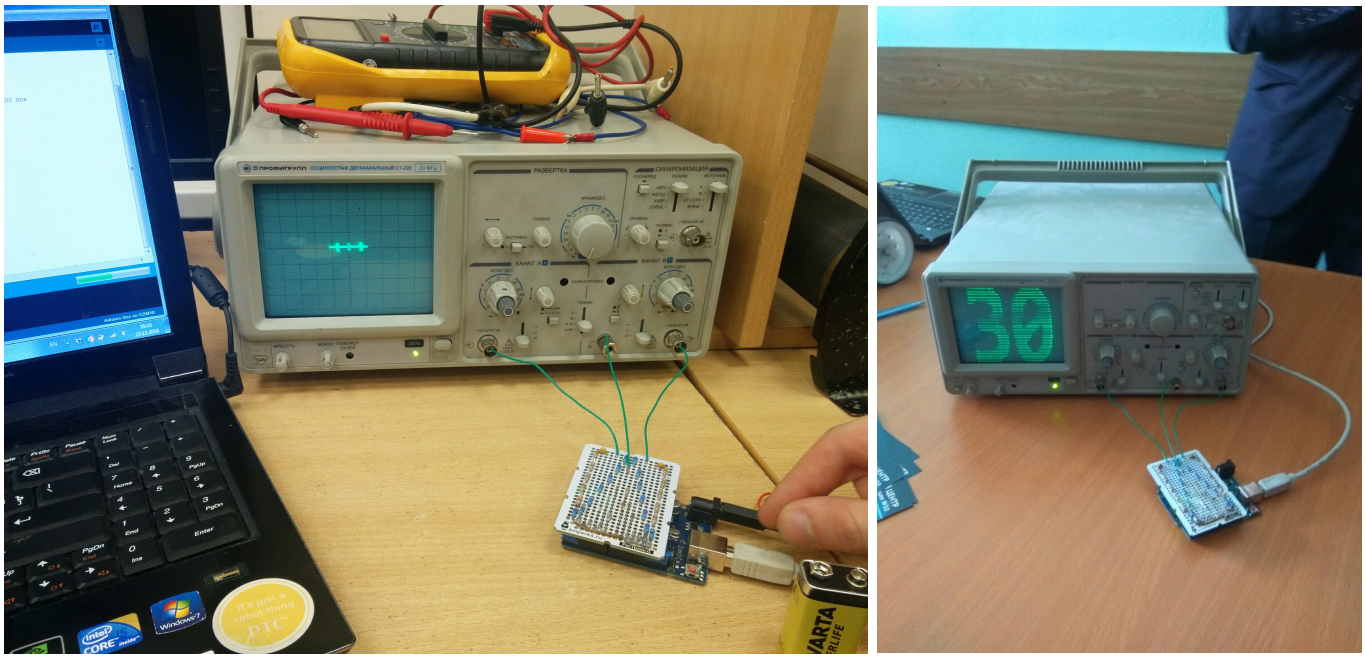


Рис. 9: Регистры процессора ATmega 168/328

При этом язык wiring поддерживает прямую работу с этими регистрами - для чтения, записи и определения режима работы портов (INPUT/OUTPUT). В то время как установка режима работы производится один раз и в этой задаче может быть написана по-простому, через pinMode(), запись в порты позволяет существенно ускорить работы программы в самом критическом и часто выполняющемся месте - подаче сигнала на ЦАПы. Вместо последовательного выполнения 6 digitalWrite(), каждый из которых содержит по дюжине строк кода, можно таким образом развернуть входной сигнал на ЦАП, чтобы биты этого числа совпадали с битами регистра, и записать результат в него (см. строки 88-89 кода 3). Это многократно ускорит выполнение этой части кода (см.Рис. 10), а одновременность записи во все порты уберёт лишние точки. Однако при использовании таких функций (не только записи в регистры, но и чтения из них) всегда нужно проверять себя: не перепутать порядок бит, не записать не туда и т.д.



а) б)

Рис. 10: Сравнение работы платы с а) первым, б) вторым способами подачи сигнала на ЦАПы

На изображениях видно, насколько ускорение работы программы увеличивает часть выводимой картинки, успевающую попасть в кадр.

4. Как уже написано выше - оба канала осциллографа должны быть в режиме ‘постоянное + переменное напряжение′, иначе ничего не работает. Кроме того, есть ощущение, что через некоторое время работы платы картинка начинает меньше дрожать (а дрожит она из-за того, что пиксель светится чуть меньше, чем нужно времени на отрисовку остальной картинки) - возможно, это происходит из-за того, что время послесвечения люминесцирующего экрана увеличивается из-за нагрева (это - предположение, возможно, такого эффекта вообще не наблюдается).

5. Чтобы вставлять русские комментарии в код программы, оформленный с помощью пакета listings в LaTeX, нужно написать:

```
\lstset{texcl=true}
```

3.3 Результат работы

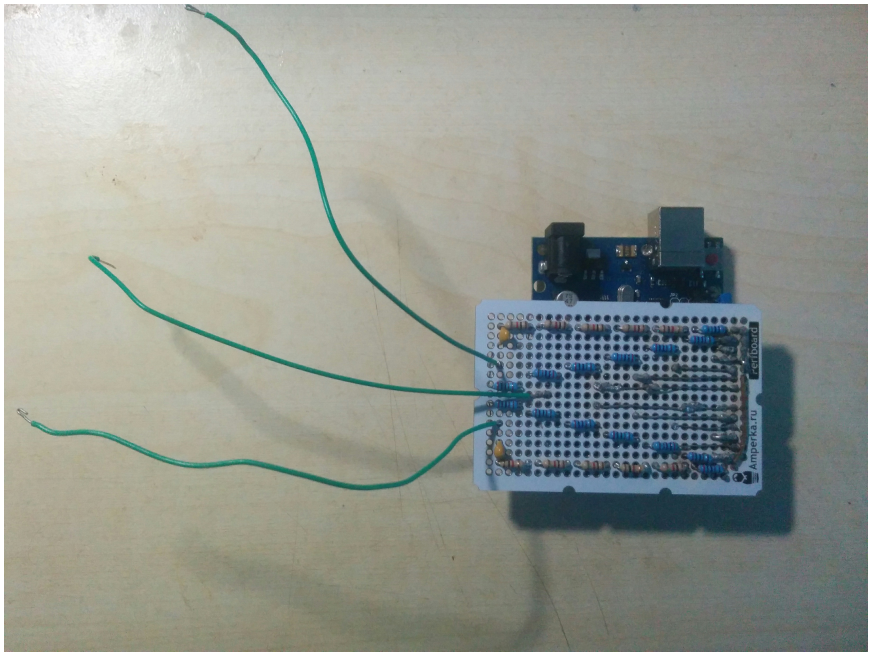


Рис. 11: Итоговый вид платы

Рис. 12: Работа платы

4 Приложение 1 - коды программ

4.1 Код на Processing v.3.2.1

```
import processing.serial.*;

Serial myPort;    // The serial port
int inByte = -1;
int i = 0;
PImage img;
int numIn = 0, numOut = 0;
int[] toSend = new int[512];

void setup()
{
   img = loadImage("IMG6.jpg");  // Загрузка картинки. Именно здесь нужно изменить путь для её смены
   img.filter(THRESHOLD, 0.5);    // Фильтр, делающий её чёрно-белой. Больше параметр - больше чёрного цвета
   img.resize(64, 64);           // Приведение к размеру 64x64
   image(img, 0, 0);        // Применение предыдущих преобразований
   print("Height is " + img.height + " Width is " + img.width);

   // Инициализация передачи по USB
   printArray(Serial.list());

   String portName = Serial.list()[0];
   myPort = new Serial(this, portName, 9600);
   int hval;
   int w;

   // Формирование массива для передачи
   for(i = 0; i < 64 * 8; i += 1)
   {
      for(hval = 0, w = 0; w < 8; w++)
      {
         if(img.pixels[i * 8 + w] != -1) // Белый цвет преобразуется только в -1 при применении фильтра
         {
            hval += (1 << (7 - w));
         }
      }

      if(i % 8 == 0)
      println("");
      print(hval);
      print(" ");
      toSend[i] = hval;
   }
   i = 0;
   println("");
   println("Transmitting...");
   delay(3000);
}

// Начало передачи
void draw() {      // draw - аналог loop на wiring, т.е. он выполняется постоянно
   background(0);
   delay(5);
   if(i < 8 * 64)
   {
```

```
55        if(i < 100) // Отладочная информация при передаче первых 100 символов
56        {
57           print("I is: ");
58           print(i);
59           print(" numIn is: ");
60           print(numIn);
61           print(" Last Received: ");
62           print(inByte);
63           print(" Last Sent: ");
64           println(toSend[i > 0 ? i - 1 : 0]);
65        }
66
67        if(i > 0 && inByte == toSend[i - 1]) // Переход к следующему элементу только после получения
              подтверждения от arduino
68        {
69           numIn++;
70           inByte = -1;
71        }
72
73        if(numIn == i) // Проверка, что получено столько же, сколько отправлено и можно продолжать
74        {
75           myPort.write(toSend[i]);
76           i++;
77        }
78     }
79
80     else if (i == 8 * 64)
81     {
82        println("Done !!!");
83        i++;
84     }
85  }
86
87  void serialEvent(Serial myPort) { // Обработка получения информации с последовательного порта
88     inByte = myPort.read();
89  }
```

Код 1: Передача изображения с компьютера

4.2 Код на wiring v.1.5.5 - приём изображения

```
#include <EEPROM.h>

int n = 0;
int val;
byte value = -1;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
   if (Serial.available() > 0)
   {
      val = Serial.read(); // Получение char'a
      Serial.write(val);   // Отправка его обратно для проверки правильности передачи
      EEPROM.write(n, val); // Запись его в память
      n++;
   }
}
```

Код 2: Приём изображения платой arduino

4.3 Код на wiring v.1.5.5 - вывод изображения

```
#include <EEPROM.h>

byte img[8 * 64] = {0};
byte lastx = 0, lasty = 0;

void pinPut(int ax, int ay) { // Функция, которая использовалась изначально для записи значений в порты
   int i = 2;
   if(ay == lasty) // Если на ЦАП y-канала уже подана нужная координата, то не меняем его
   {
      for(; i < 8; i++) // Проходим по циклу все биты x-координаты и записываем их в порты ЦАПа x-канала
      {
         if (ax & 1)
         {
            digitalWrite(i, HIGH);
         }
         else
         {
            digitalWrite(i, LOW);
         }
         ax = ax >> 1;
      }
      return;
   }
   for ( ; i < 8; i++) // То же самое, но уже для обоих каналов - если y-координата изменилась
   {
      if (ax & 1)
      {
         digitalWrite(i, HIGH);
      }
      else
      {
         digitalWrite(i, LOW);
      }
      ax = ax >> 1;
      if (ay & 1)
         digitalWrite(15 - i, HIGH);
      else
         digitalWrite(15 - i, LOW);
      ay = ay >> 1;
   }
   return;
}

char rotate(char a) { // Вспомогательная функция для второго способа записи в порты - "переворот"первых шести битов числа
   char c = 0;

   for (char i = 0; i < 6; i++)
   {
      c += (a & 1) << (5 - i);
      a = a >> 1;
   }
   return c;
}


void setup() {
   int i = 2;
```

```
58    for(; i <= 13; i++)
59       pinMode(i, OUTPUT);
60
61    int x = 0, y = 0, address = 0;
62    byte val;
63    for(; y < 64; y++) // Загрузка картинки в переменную, чтобы не читать постоянно из EEPROM
64    {
65       for(x = 0; x < 8; x++)
66       {
67          img[y * 8 + x] = EEPROM.read(address);
68          address++;
69       }
70    }
71 }
72
73 void loop() {
74    char x, y, n;
75    char addrX;
76    byte val;
77    while (1) // такой цикл работает чуть быстрее, чем loop()
78    {
79       for (y = 0; y < 64; y++)
80       {
81          for (x = 0, addrX = 0; x < 8; x++, addrX += 8)
82          {
83             val = img[y * 8 + x];
84             for (n = 7; n >= 0; n--) // Проход по битам элемента картинки, каждый бит - одна точка
                   на экране
85             {
86                if (val >> n & 1)
87                {
88                   PORTD = (addrX + (8 - n)) * 4; // Прямая запись в регистры портов, второй способ
89                   PORTB = rotate(y); // Числа изменяются, чтобы их биты совпали с нужными пинами
90                }
91             }
92          }
93       }
94    }
95 }
```

Код 3: Вывод изображения платой arduino