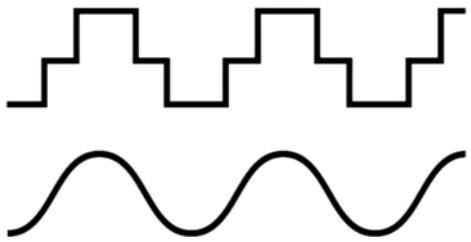
Arduino Oscilloscope



Nicu-Serban POP

Polytechnic University Timisoara

1. Introduction

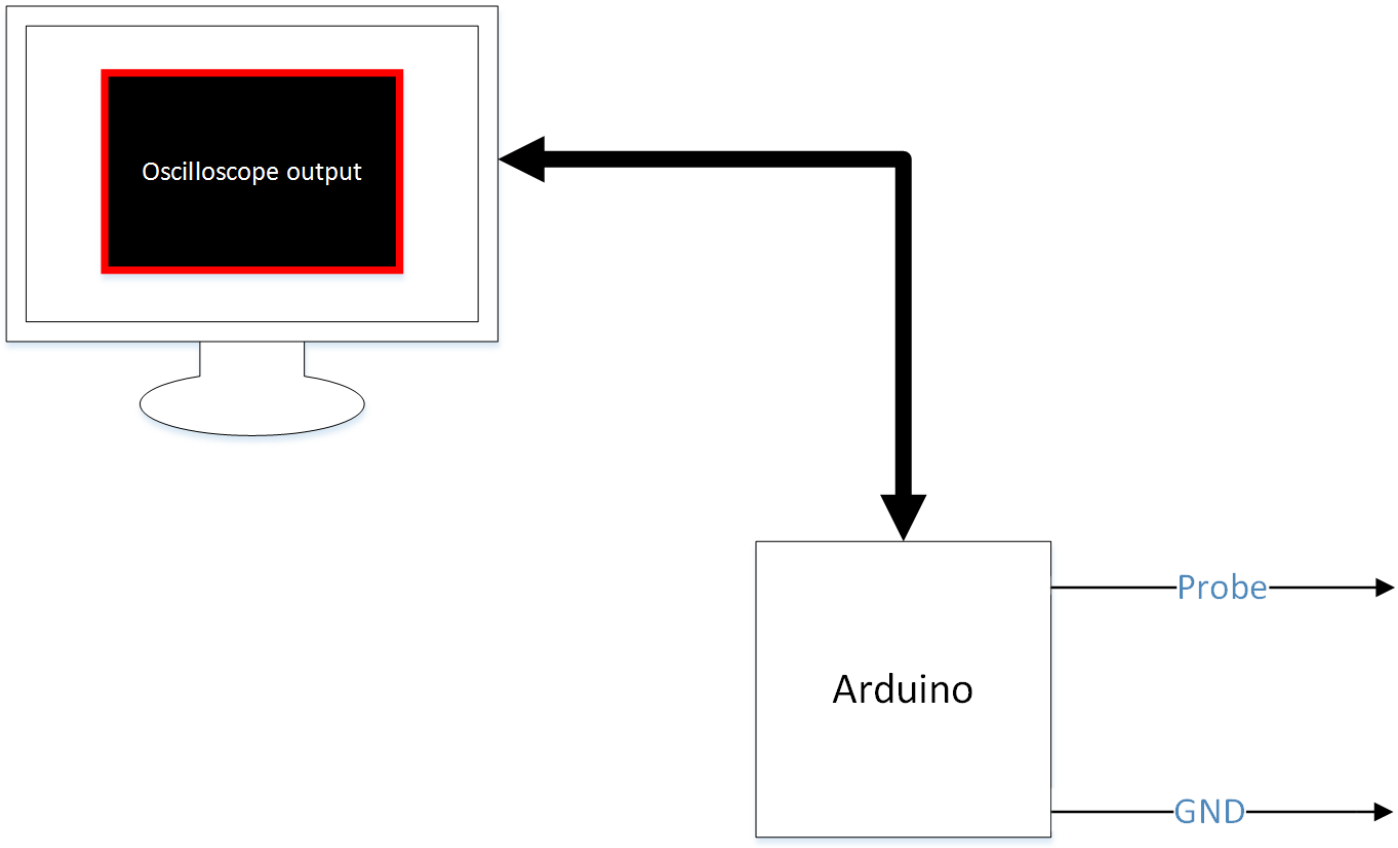
The application is an Arduino based oscilloscope for measuring and testing different kinds of circuits.

What is an oscilloscope? An oscilloscope is a type of electronic test instrument used by engineers and technicians to test, verify and debug electronic circuits and allows us to:

- Convert electrical input signals into visible trace on a screen (convert electricity into light).
- Dynamically graph time-varying electrical signals in two dimensions (usually voltage vs. time).

2. Architecture

The platform used for this project is a Arduino Uno but can be transposed to any type of Arduino. Power is supplied via the USB 5V cable used for connecting the Arduino to the computer. The Arduino uses only one analog pin (as the probe) and one of the grounds to be connected to the ground of the circuit we want to test. The schematic is as follows:



The Arduino board contains a 6 channel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit.

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second. The sample frequency with analogRead is approximately 10kHz.

To measure an analog signal there has to be a voltage level to compare it with. This voltage is called the reference. In the ATMEGA328 of the Arduino this reference voltage is also the maximum voltage that can be measured. The voltages are always measured relative to the ground.

3. Arduino code explained

```
void setup()
{
  Serial.begin(115200); //higher baud rate means faster speed
}

void loop()
{
  int data = analogRead(0); // analog 0 pin
  Serial.write( 0xff ); // first byte
  Serial.write( (data >> 8) & 0xff ); // split into two bytes
  Serial.write( data & 0xff );
  delay(0);
}
```

The analogRead function returns a ten-bit value, but that won't fit into a single byte so, in order for us to send it to the computer, we split it into two bytes plus a marker byte. The high byte of the result is transferred first so the first byte after a sequence of 0xFF is always the high part of the value.

4. Processing code explained

Following will be explained how the Processing code works. More information on what every line of code does can be found on the code itself as a comment.

- void setup()

This functions sets and initializes the values the program is going to use during execution. We set the size of the screen (smaller screen means less to compute hence the lower the delay) and initialize the port for the communication protocols.

- int convertToPixels(int val)

This function converts the value it is called by to a value in concordance to the resolution we set.

- int readData ()

Here we merge the two received bits into one integer while the number of bytes available is greater or equal to three (*the marker byte and the two split bytes*) and return the value.

- void buildArray(int value)

This function puts all the values it receives from the Arduino into an array for latter use.

- void lineDraw()

This is a function used for drawing the lines for the oscilloscope. First, we set the color we want the lines to be. Then, we draw lines between point (x0,y0) to (x1,y1).

- void referenceLines()

Here is where all the horizontal lines splitting the screen and the text on the right is drawn. The process in doing so is very simple, we just set the color we want to use and draw a line from the left-most point and the desired height value to the right-most point and the desired height. For the text we need to set the color, the size and the (x,y) coordinates.

- void keyPressed()

Saves a screenshot of the window with all the data drawn on it. Overwrites old files.

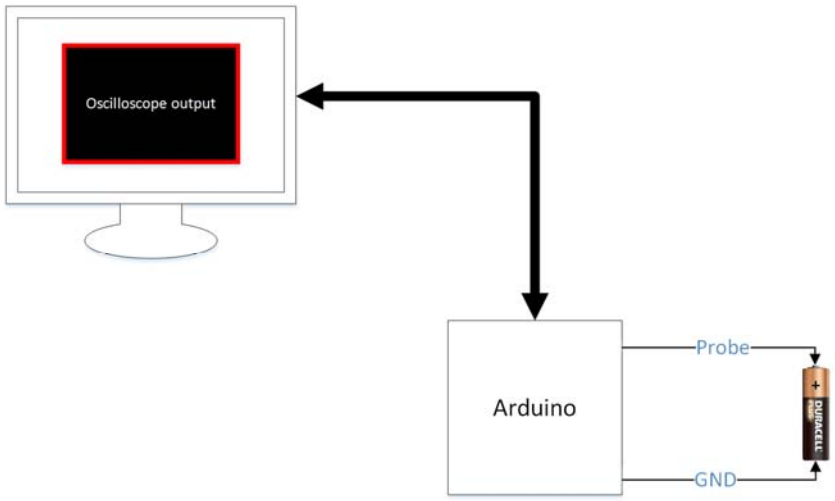
- void draw()

This is where all the functions are initialized for drawing the grid, the lines and the text.

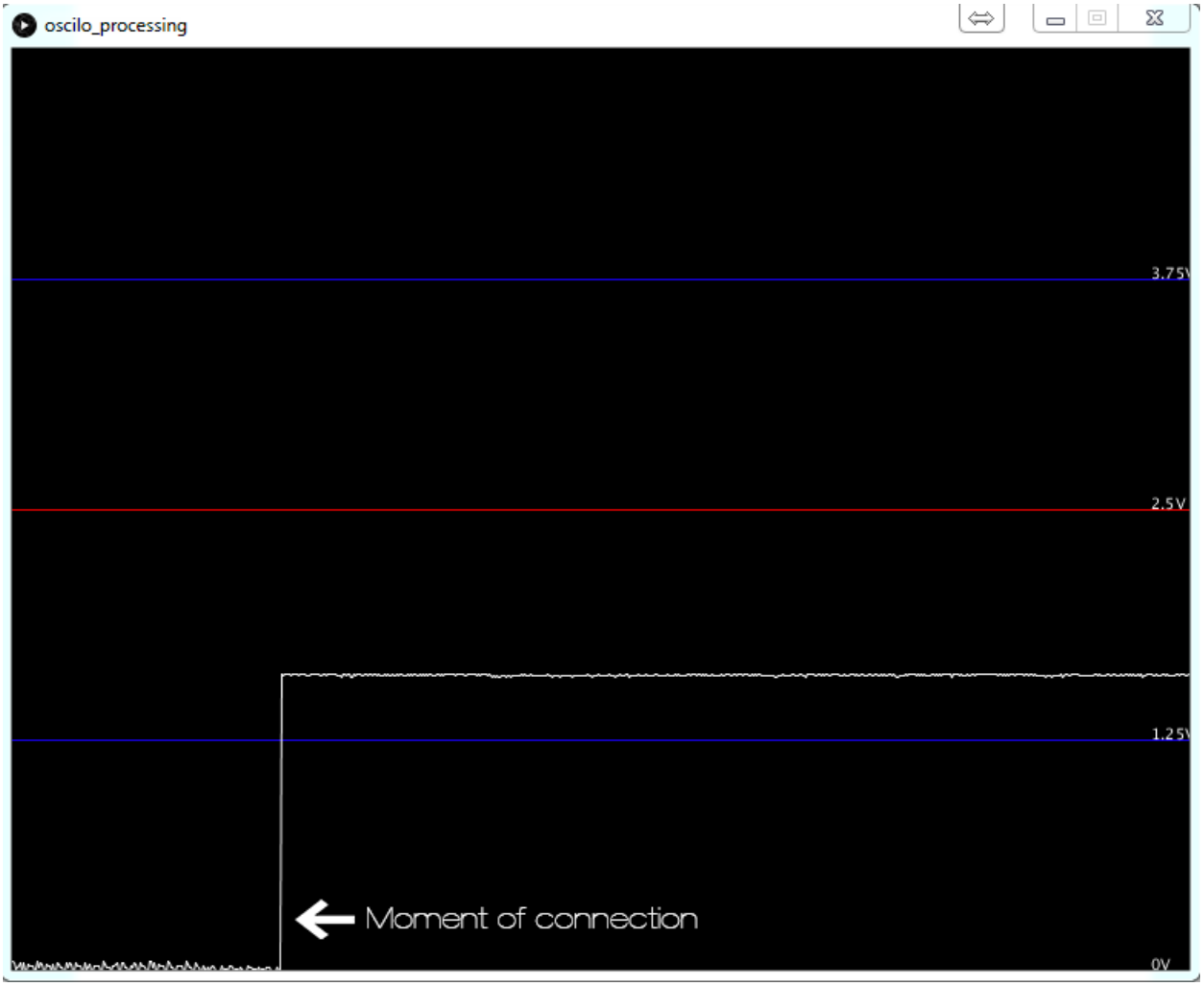
5. Example for usage
   - Simple battery tester.

Warning!!: Do not use batteries and circuits larger than 5V.

Connection Schematic:



Oscilloscope output:



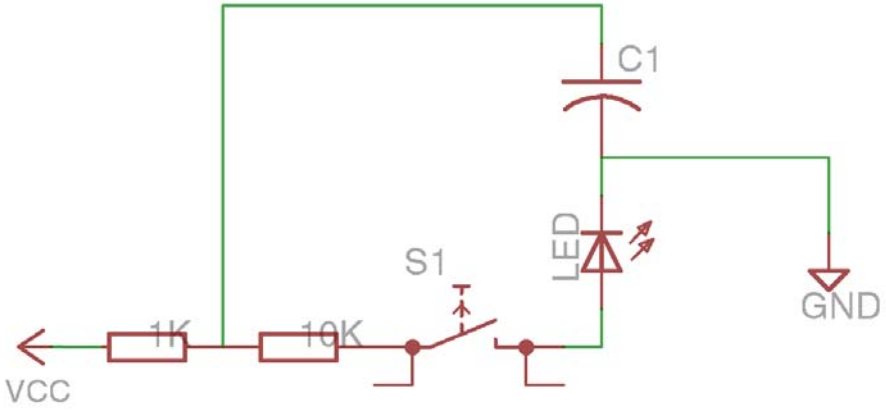
Here you can clearly see the moment the probe connects to the battery and outputs it's voltage.

- Capacitor charging and discharging.

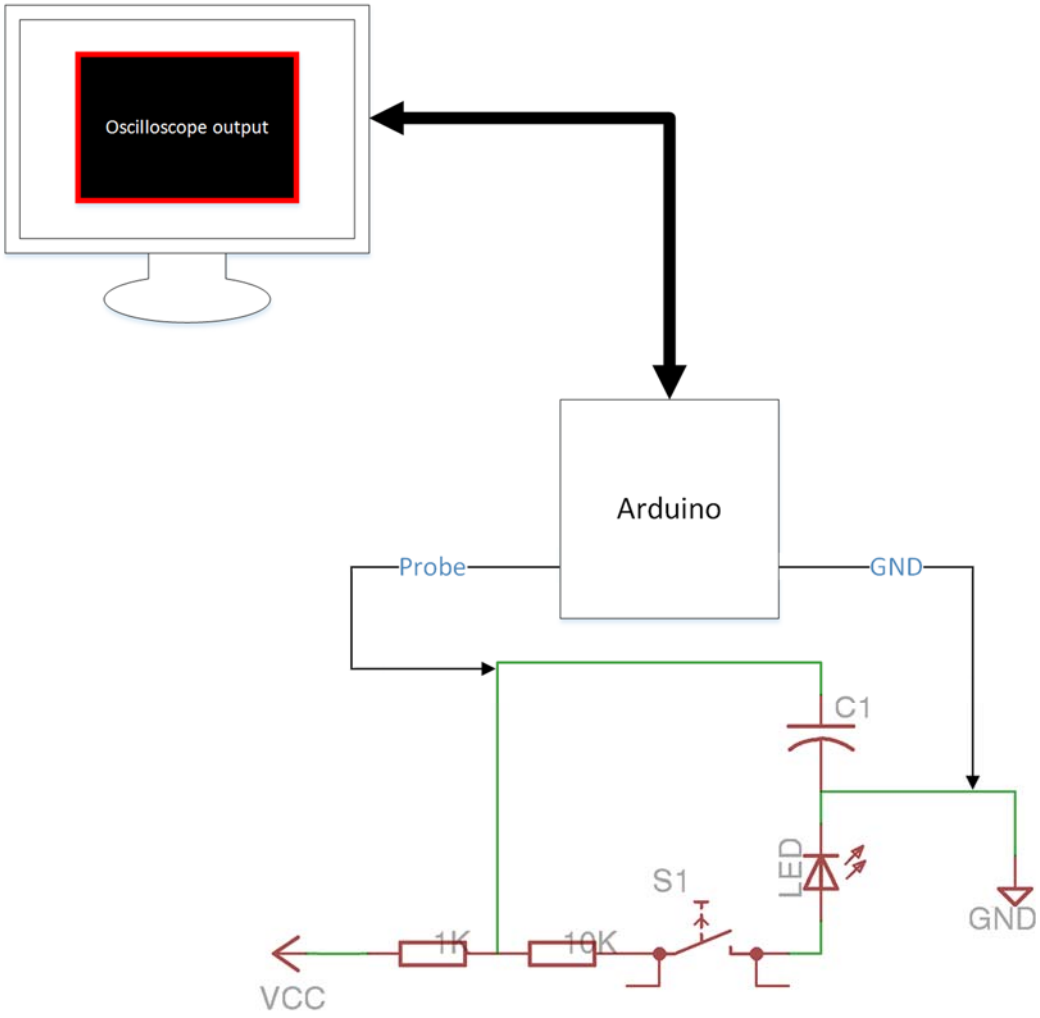
The following circuit shows how a capacitor charges and discharges. The capacitor is charged from the Arduino and every time the pushbutton is pressed, the circuit closes and the capacitor will discharge.

Note: the capacitor will discharge only to about 2.8 volts. This is because of the 1K resistor.

Circuit:



Connection schematic:



Oscilloscope output: