AtTiny LED control print

2019-01-30

JelleWho

Table of contents

Summary

This document is made to guide the user of the “AtTiny Led Control PCB” through the process of using the product. Although this document isn’t necessary to read before use, it’s recommended to do so at least once during initial setup.

I will explain shortly how to quickly get it up and running. But I will also talk about how to edit or change a few things, just enough to get you started. I also referenced the most common mistakes made that deserve your attention. finally I included a current copy of the schematic, PCB and the code.

This product and the codes are provided "as is", without warranty or any kind, express or implied, including but not limited to the safety of the code and the product. In no event shall I (JelleWho) be liable for any claim, damages, or other liability, no matter the connection or contracts between us. Feel free to use the product and code and comment and work on it, and don’t forget to credit where credit is due.

1. Quick start guide

Figure 1 The PCB is a render of the PCB, on the left side of this PCB are the 8 connections as follows,

##	Description	Voltage range
I1	Input 1 (red)	0-24V*
I2	Input 2 (green)	0-24V*
I3	Input 3 (blue)	0-24V*
I4	Input 4 (Animation 1)	0-24V*
I5	Input 5 (Animation 2)	0-24V*
Q1	Output 1 (data line LED strip)	5V out
GND	The ground connection	0V
5V	5V VCC connection	5V

*If R1-5 are the default values

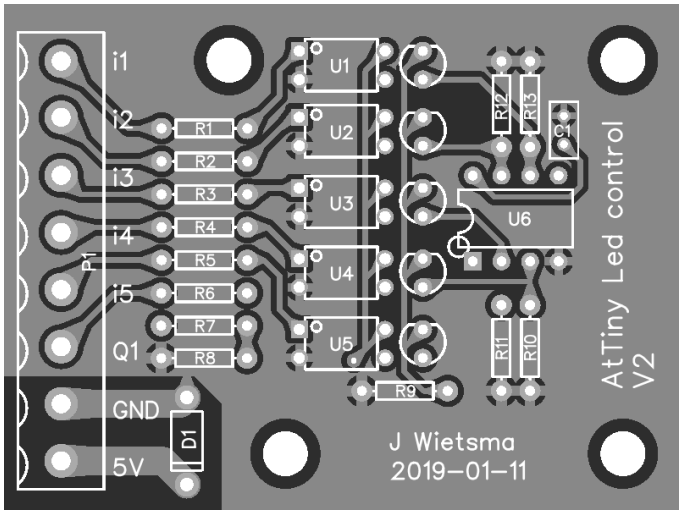


Figure 1 The PCB

The LED strip (WS2812B) has 3 connections they are as follows;

##	Description	Voltage range
GND	The ground connection	0V
Di	Input (Data line LED strip)	0-5V
5V	5V VCC connection	5V

It's recommended to use a 1 Meter 60Led/M WS2812B strip, since this has been setup in the code.

1.1. How to change

1.1.1 Input voltage

By default, the input pins are configured to 24V. This is done by the use of R1-5 respectively. The PC817 optocoupler has an input current of 0.050A and a drop voltage of 1.2V. When taking the default 24V input voltage, this equation for resistors 1 to 5 is obtained:

$$(24V-1.2V)/0.050A = 456\Omega$$

If another Input voltage is desired the values need to be changed accordingly. This can also be done if the optocoupler needs to be changed (for example if you want to use a TLP621 which has a 1.15V drop and 0.0016A current). This change does not affect the 5V VCC supply voltage!

1.1.2 Firmware

To program the AtTiny a programmer is required. For example you can use an Arduino to program the chip using this PCB [1] this one has been verified to work and can be bought if you contact me.

The Arduino-ish firmware for this chip can be found on the Github page [2].

1. Download and install the Arduino software.
2. Download the AtTiny plugin [3] and the FastLED plugin [4]
3. Program the Arduino as ISP (see example sketch) and connect the AtTiny-programmer PCB.
4. Set the processor to be AtTiny85 at an internal 16MHz clock and 'Arduino as ISP' as programmer and burn the bootloader while the PCB is in programming mode and the AtTiny is connected.
5. Now open and upload the Arduino-AtTiny-Led-controller firmware sketch [2].
6. Unmount the AtTiny and mount it to the LED PCB, pin 1 is marked with the dot/circle.

1.2.3 Animations and settings

In the firmware mentioned above [2] you can edit the values, the section below "Just some configurable things'' can be easily edited without much knowledge. These include things like number of LEDs, which is capped at a hardware max of 120 but is not recommended since it would have low memory space (Default it's set to 60).

1.2. Common mistakes

- **Overvolt** Do NOT connect more than 5V to the 5V connection. This will result in permanent destruction. Only the inputs 1-5 are made for 24V (if R1-R5 are set as default).
- **Reverse current** Do NOT connect the GND and 5V in reverse, on the PCB the diode will hopefully only blow. And the LED strip will blow the firs led permanently.
- **Shifted ground** Make sure to connect the LED strip ground to the ground of the controller.
- **Under Power** each color in a LED used 60mA, there are 3 colors and there 60 LED's. this concludes to a total power draw of 10.8A on the 5V rail (If the LED's are full bright white and on). The AtTiny doesn't draw enough power to be calculated here (<5mA). If for example only a red blink is used this would calculate to 60mA * 60 LEDs * 50% duty cycle = 1.8A and this could be powered from a power bank or such.

2. References

[1] https://easyeda.com/jellewietsma/ATTiny-programmer

[2] https://github.com/jellewie/Arduino-Attiny-LED-Controller

[3] https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

[4] https://github.com/FastLED/FastLED/releases

[Figure 1] By JelleWho from EasyEda, The PCB, retrieved on 2019-01-14

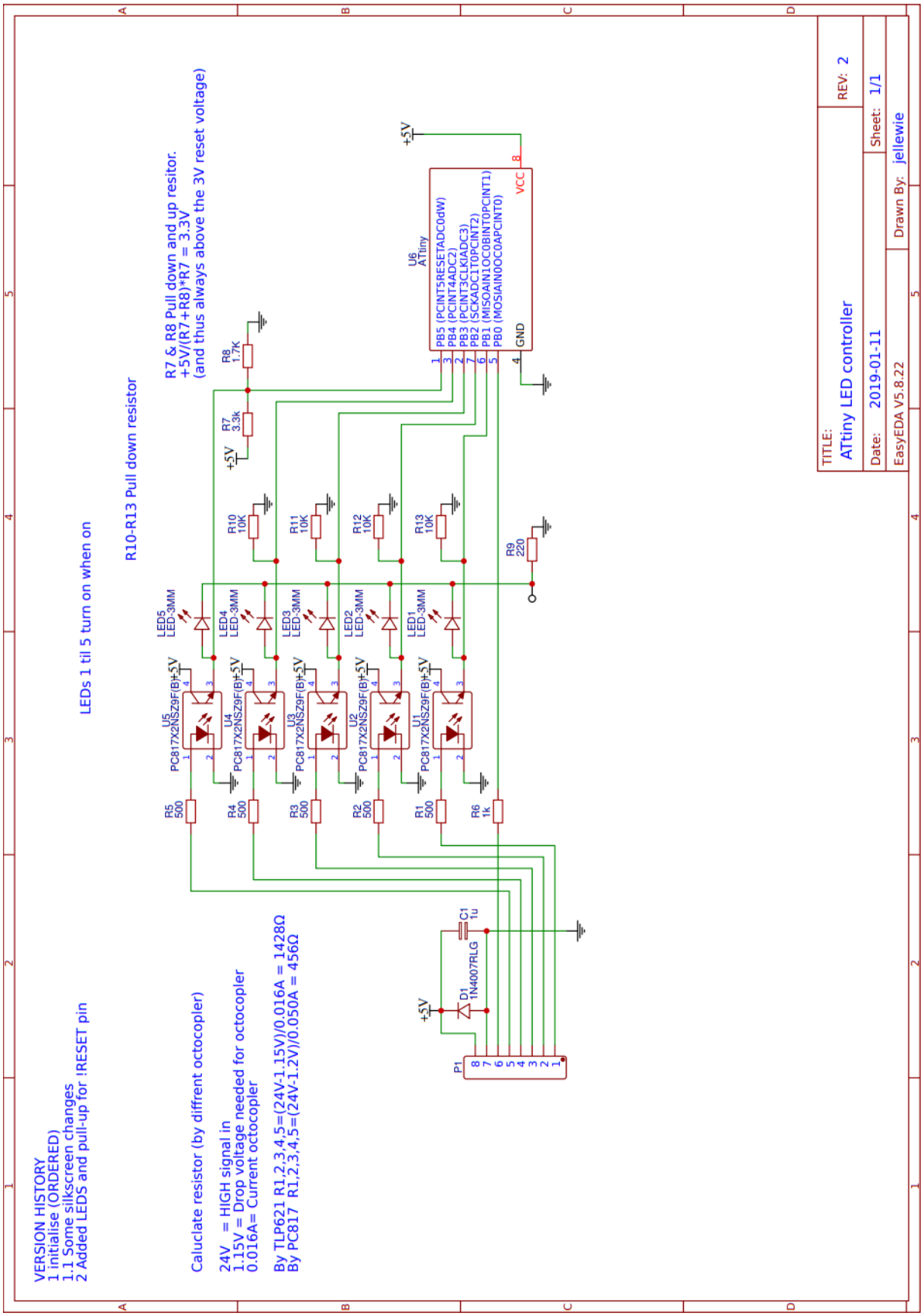
[Appendix 3.2] By JelleWho from EasyEda, PCB top side, retrieved on 2019-01-14

[Appendix 3.3] By JelleWho from EasyEda, PCB bottom side, retrieved on 2019-01-14

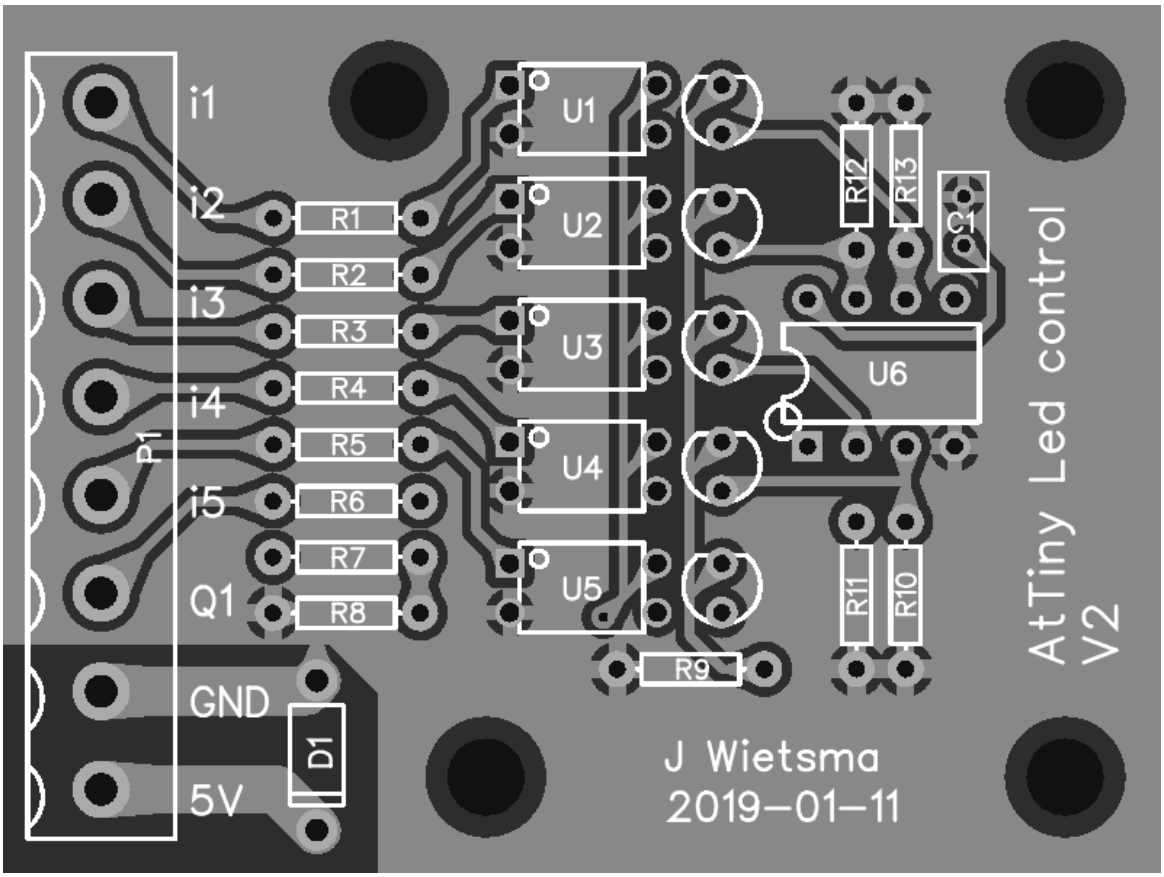
[Appendix 3.4] By JelleWho from GitHub, The code, retrieved on 2019-01-14

3. Appendix

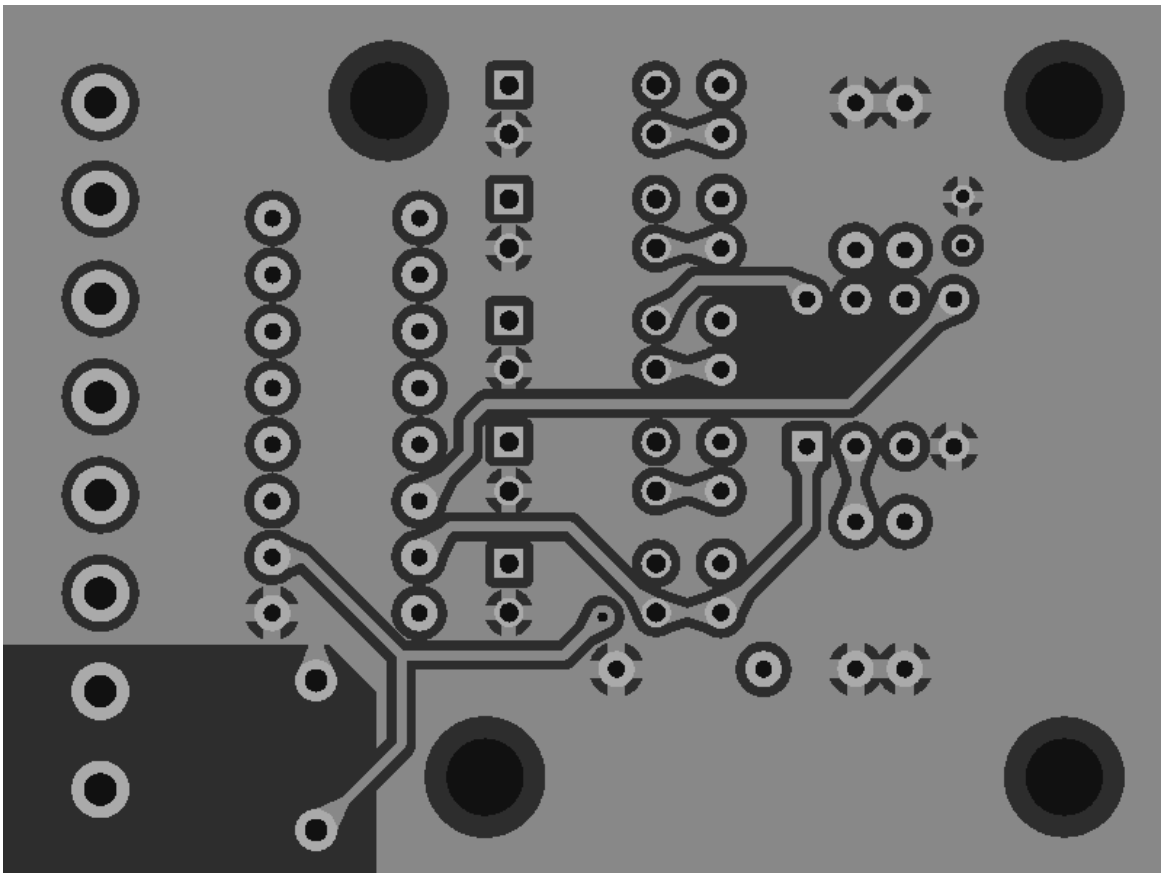
3.1. The Schematic



3.2. PCB Top side



3.3. PCB Bottom side



3.4. Firmware

```
/*
Program written by JelleWho for a standalone chip that takes 5 digital inputs and convert them
to a FastLed controlled led strip. See https://github.com/jellewie/Arduino-Attiny-LED-
Controller
*/
//Just some configurable things
const static int TotalLEDs = 100;  //The total amounts of LEDs in the strip
const static byte DelayAnimationBlink = 50;  //Delay in ms for an update
const static byte DelayAnimationFlash = 250; //Delay in ms for an update
const static byte DelayAnimationMove = 20;   //Delay in ms for an update
const static byte Brightness = 255;//The brightness of the LEDs
const static byte MoveAmount = 2;  //Quantity of the sections in the move animation
const static byte MoveLength = 5;  //Length of the sections in the move animation

static const byte PAO_LED = 0;
static const byte PDI_Red = 1;
static const byte PDI_Green = 2;
static const byte PDI_Blue = 3;
static const byte PDI_Blink = 4;
static const byte PDI_Flash = 0;
#include <FastLED.h>
CRGB LEDs[TotalLEDs
void setup()
  pinMode(PAO_LED,OUTPUT);
  pinMode(PDI_Red,INPUT);
  pinMode(PDI_Green,INPUT);
  pinMode(PDI_Blue, INPUT);
  pinMode(PDI_Blink,INPUT);
  pinMode(PDI_Flash,INPUT);
  FastLED.addLeds<WS2812B, PAO_LED, GRB>(LEDs, TotalLEDs);
  FastLED.setBrightness(Brightness);
  fill_solid(&(LEDs[0]), TotalLEDs, CRGB(0, 0, 0));
  FastLED.show();
}
void loop() {
  static byte Mode = 0;
  bool UpdateLEDs = false;
  byte Red = 0;
  byte Green = 0;
  byte Blue = 0;
  if (digitalRead(PDI_Red) == HIGH)
    Red = 255;
  if (digitalRead(PDI_Green) == HIGH)
    Green = 255;//^^
  if (digitalRead(PDI_Blue) == HIGH)
    Blue = 255;
  if (digitalRead(PDI_Blink) == HIGH) {
    if (analogRead(PDI_Flash) >= 900) {
      if (Mode != 3) {
        Mode = 3;
        fill_solid(&(LEDs[0]), TotalLEDs, CRGB(0, 0, 0));
        UpdateLEDs = true;
      }
      static int CounterMove = 0;
      int OffsetMove = TotalLEDs / MoveAmount;
      int poss[MoveAmount];
      if (CounterMove >= TotalLEDs) {
        CounterMove = 0;
    } else {
```

```
      EVERY_N_MILLISECONDS(DelayAnimationMove) {
       CounterMove++;
       fill_solid(&(LEDs[0]), TotalLEDs, CRGB(0, 0, 0));
       UpdateLEDs = true;
      }
    }
    for (int i = 0; i < MoveAmount; i++) {
    poss[i] = CounterMove + (OffsetMove * i);
    int posX;
    if (poss[i] >= TotalLEDs) {
      posX = poss[i] - TotalLEDs;
    } else
      posX = poss[i];
      if (posX <= (TotalLEDs - MoveLength)) {
        fill_solid(&(LEDs[posX]), MoveLength, CRGB(Red, Green, Blue));
      } else if ((posX >= (TotalLEDs - MoveLength)) && (posX <= TotalLEDs)) {
        int calc1 = (TotalLEDs - (posX + MoveLength)) * -1;
        int calc2 = MoveLength - calc1;
        fill_solid(&(LEDs[posX]), calc2, CRGB(Red, Green, Blue));
        fill_solid(&(LEDs[0]) , calc1, CRGB(Red, Green, Blue));
      }
    }
  } else {
    if (Mode != 1) {
      Mode = 1;
      fill_solid(&(LEDs[0]), TotalLEDs, CRGB(0, 0, 0));
      UpdateLEDs = true;
    }
    EVERY_N_MILLISECONDS(DelayAnimationBlink) {
      UpdateLEDs = true;
      static byte CounterBlink = 0;
      static byte LengthBlink = TotalLEDs;
      static byte AlwaysOn = 1;
      fill_solid(&(LEDs[0]), LengthBlink,CRGB(0, 0, 0));
      fill_solid(&(LEDs[0]), AlwaysOn, CRGB(Red, Green, Blue));
      fill_solid(&(LEDs[0]), CounterBlink, CRGB(Red, Green, Blue));
      CounterBlink++;
      if (CounterBlink > LengthBlink)
       CounterBlink = 0;
    }
  }
} else if (analogRead(PDI_Flash) >= 900) {
  if (Mode != 2) {
    Mode = 2;
    fill_solid(&(LEDs[0]), TotalLEDs, CRGB(0, 0, 0));
    UpdateLEDs = true;
  }
  static bool FlashStateOn = false;
  EVERY_N_MILLISECONDS(DelayAnimationFlash) {
  FlashStateOn = !FlashStateOn;
  if (FlashStateOn)
    fill_solid(&(LEDs[0]), TotalLEDs, CRGB(Red, Green, Blue));
  else
    fill_solid(&(LEDs[0]), TotalLEDs, CRGB(0, 0, 0));
  UpdateLEDs = true;
  }
} else {
  Mode = 4;
  fill_solid(&(LEDs[0]), TotalLEDs, CRGB(Red, Green, Blue));
  UpdateLEDs = true;
}
if (UpdateLEDs)
FastLED.show();
}
```