Worm Tracker Documentation

Contenido

[Introduction 3](#_Toc68531325)

[Software 3](#_Toc68531326)

[Hardware 3](#_Toc68531327)

[Arduino (in general) 3](#_Toc68531328)

[Temperature sensor 4](#_Toc68531329)

[Potentiometer (manual) 4](#_Toc68531330)

[Dimmable high power leds 5](#_Toc68531331)

[TEC control 5](#_Toc68531332)

[Fan control 6](#_Toc68531333)

[Stepper motor control 7](#_Toc68531334)

# Introduction

# Software

# Hardware

An arduino is a microprocessor that is able to perform simple tasks like turning on/off switches and measuring voltages. We will use this to precisly control three high power leds for illumination, a Peltier element (thermoelectric couple) and a couple of fans for temperature control and, a motor driver to digitally control the focus of the camera. Also, we use the Firmata protocol to establish a robust communication between the Arduino and a computer. In the following sections you can find a detailed description of why and how the controller board was designed.

### Arduino (in general)

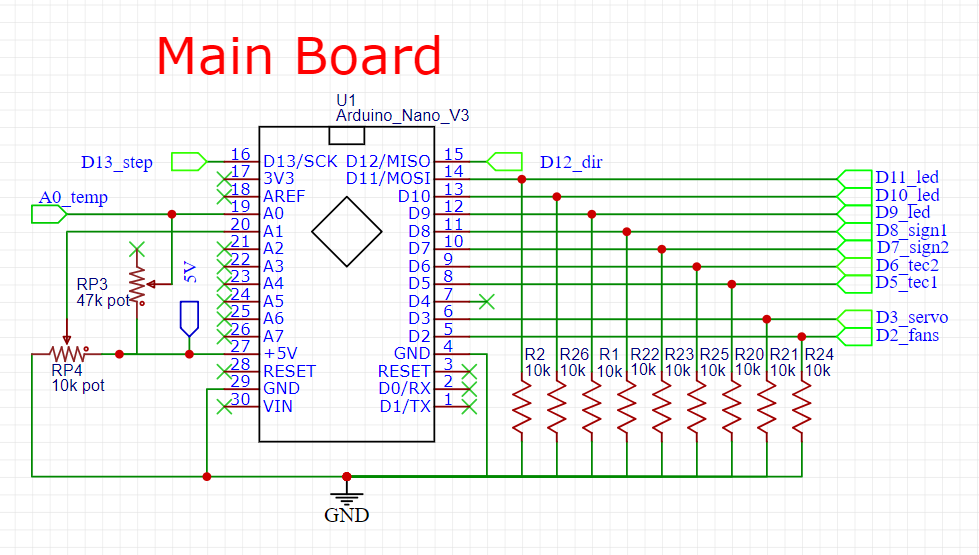
We use an Arduino Nano board, because it has enough digital pins to control everything we need, it can be based on either ATmega168 or ATmega328. In general, it features several modules that will be described in the following sections namely: thermistor, manual potentiometer, led lighting, temperature control, stepper motor control and signaling leds (see Table X)

**Pulldown resistors.** When the Arduino is turned on, all pins are turned as input by default. During this time, digital pin values are floating and are neither high nor low. After a few seconds, all pins are set to output low [DO IN SOFTWARE]. To avoid this transient response, we have added to every digital output pin a pulldown resistor to keep this from happening.

**<W>** The Arduino board must necessarily share the ground connection with the external power supply at all times.

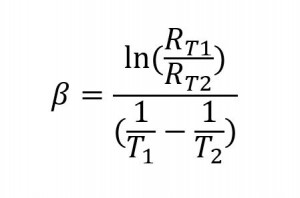
|  |  |  |  |
| --- | --- | --- | --- |
| Number | Function | Pins | Comments |
| 1 | Temperature sensor | A0 | Analog thermistor |
| 1 | Potentiometer | A1 | Manual action |
| 3 | Dimmable high power leds | D9, D10 and D11 | 1W, 350mA |
| 1 | One-way TEC control (60W) | D5 | w/ MOSFET |
| 1 | Two-way TEC control (60W) | D5 and D6 | w/ motor driver |
| 1 | Fan control | D2 | Two fans in parallel |
| 1 | Stepper motor control | D12 and D13 | Used to focus |
| 1 | Signaling leds | D7 and D8 | Bicolor |

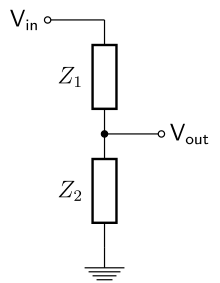
Table 1. Main board functions



### Temperature sensor

We use a thermistor called an “3950 NTC Thermistor Temperature Sensor 10K 1%” with a beta parameter equal to 3380K +/- 1%. This device is a resistor whose value depends strongly on its temperature. As it is an N(egative)TC, when the temperature raises, the resistance falls. To compute the temperature we need two equations: one to compute the temperature of the probe from its resistance, and another to compute the resistance from an analog voltage (as measured by the Arduino).

First, the equation to the left can be rearranged to provide a resistance value (R\_T2) given a temperature (T2) and a reference resistance (R\_T1) at a reference temperature (T1). Note that temperatures are in Kelvin. The reference temperature T1 is usually 25ºC, or 298K. Its resistance at such temperature R\_T1 is about 10kOhm, and the beta constant is said to be 3380 1/K. Plug everything into the equation and you can obtain T2 as a function of R2.

To measure R2, we use a voltage divider as the schematic in the left. Vin is a constant reference voltage (5V in our case), Z1 and Z2 are two resistors and Vout is the voltage measured at that point (A0 in our case). Either Z1 or Z2 can be a reference resistor and the other is the thermistor (as a variable resistor, R2 in the previous paragraph). The voltage measured in Vout, will be: Vout = Vin\*(Z2/[Z1+Z2]). From this equation, one can obtain a function that returns the variable resistance R2 as a function of the measured voltage Vout. Depending on your reference resistor, temperature will be measured more accurately in a certain range of temperatures. We recommend a reference resistance equal to the NTC resistance at around 20ºC to maximize precision at this temperature. On top of this, Arduino does not directly measure Vout, but rather, the fraction Vout/Vin in 8bit. This means that when reading A0=0 then Vout=0V and when A0=256 then Vout=Vin. We finally have:

In our case we have used the scenario where Z1 is a 47kOhm variable resistor and Z2=R2. With this, we can adjust Z1 for improved temperature accuracy.

Note that the previous is a general procedure, but in our case, as we use the Firmata protocol to es

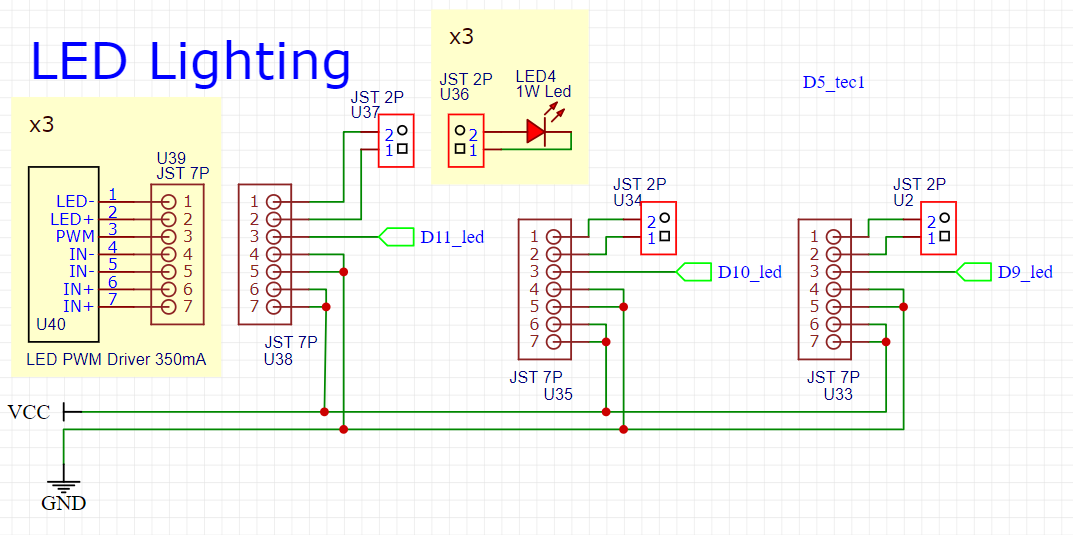
### Potentiometer (manual)

We have added a potentiometer that can be turned manually to make it easy to adjust the output of any pin. The goal was to easily dim the principal led, but other functions can be set in the myArduino class. The working principle is that of the voltage divider as in the previous section, in this case however Z1+Z2 is fixed to the potentiometer’s maximum value (10kOhm in our circuit).

### Dimmable high power leds

The Arduino board cannot output over 40mA in a single pin, hence, to drive high power leds that require 350mA we need a led driver. We use a cheap module that allows to easily dim the led’s brightness with a PWM signal. A PWM signal is similar to an analog voltage that can take a continuum of values, in this way, the higher the voltage, the more intense the lighting.

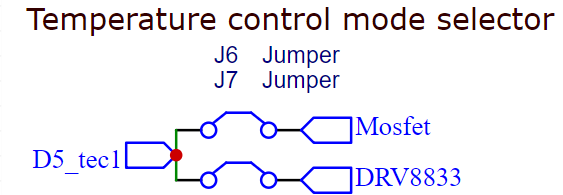
For each high power led, we use either digital pins D9, D10 or D11. We have included a 4.7 kOhm resistor to limit the current provided by the Arduino to about 1mA. Also, we have tuned the fermata script to increase the frequency of the PWM signals driving the leds to avoid resonance with the camera FPS.



### TEC control

To control the temperature inside the chamber we use a “thermoelectric couple”, or TEC, also known as “Peltier element”. This device can cool down one of its faces and heat up the other while current is flowing. If the heat of the hot side is well dissipated, then the cold side can reach very low temperatures.

**<W>** It is very important that there is thermal paste between the TEC and both heat dissipators.

In our case we use a Peltier element rated max at 12V or 5A. The measured resistance is indeed about 2.5 ohm, but it can vary largely with temperature (or current). There is no easy and perfect way to control a TEC, so we propose two options: a one-way control that is only able to cool down the chamber (to hopefully very low temperatures) and a two-way control that is also able to heat up the chamber.

**<W>** To select between one- or two-way modes, we added a pair of jumpers. ONLY one should always be connected.

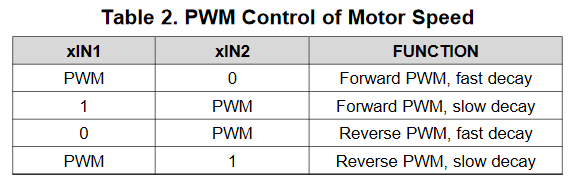
#### One-way control of the TEC

To drive the TEC at different voltages we will use a logic-level, switching MOSFET. By doing this, we can apply a PWM signal to the MOSFET’s gate to quickly turn it on/off so that the TEC sees a fraction of the total input voltage. If the duty cycle is set to 100% then around 4.5A will be going through the TEC providing a large cooling power but not efficiently! Recall that resistive power losses go as the current squared.

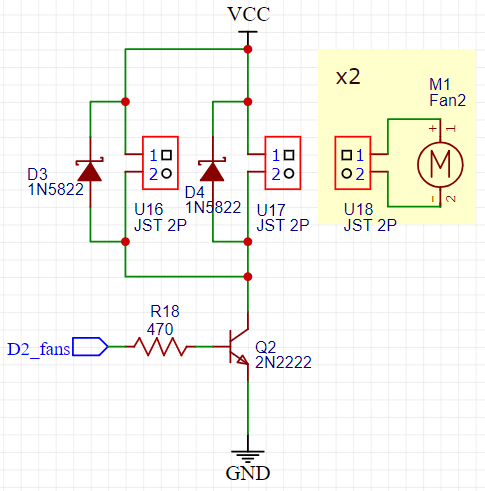
We chose the IRLZ44 due to its large voltage headroom (60V) for possible switching spikes and for its low R\_DS=28mOhm. In the worst case scenario, 5A will go through the MOSFET, which will need to dissipate <1W. Even so, we can add a TO-220 dissipator. In the circuit we simply added a 470 Ohm resistor between the Arduino pin and the gate to limit the current to 10mA (and to protect the Arduino from back emf during switching). As the TEC is a resistive load, we shouldn’t need a flyback diode, however we attached one just in case parasitic inductance causes problems. We also added a decoupling cap between the gate and the ground.

#### Two-way control of the TEC

To drive the TEC in both cold and hot modes, we must use the H-bridge of a motor driver. We used DRV8833 because of its relatively high output power (up to 1.5A per channel, so a total max of 3A which is less than the 4.5A of the one-way MOSFET). The module already includes different types of protection (overtemperature, overcurrent…) so the pinout is pretty straightforward. Depending on the IN1 and IN2 values cooling/heating modes are toggled.



### Fan control

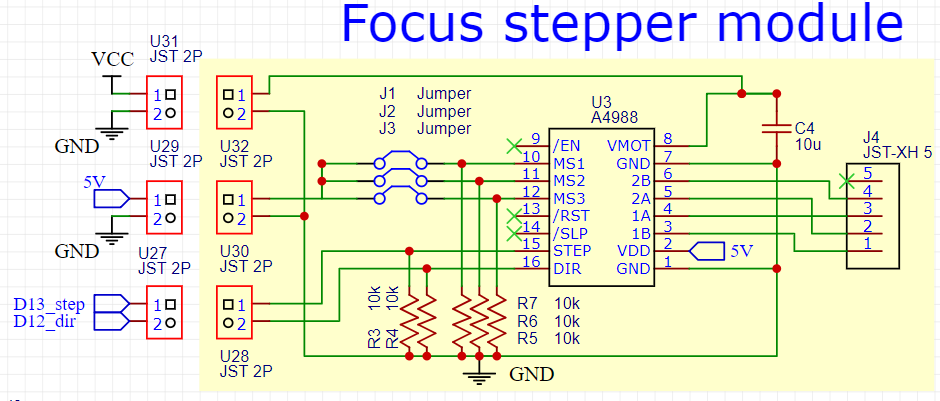
There should be TWO fans, one for the hot side and another for the cold side because i) the fan on the cold side will improve cooling power to the chamber, and ii) the fan on the cold side might be needed if the TEC direction is reversed and it starts heating the chamber.

Both fans are controlled with a 2N222 transistor as the necessary current is about 400mA (300mA from the large fan and an extra 100mA from the smaller one inside the chamber). Although fans have supposedly built-in flyback diodes, we added a pair of 1N5822 Schottkys (although they won’t be PWMed) for back EMF protection. There is also a 470 Ohm between the Arduino pin and the transistor’s base to limit the current to 10mA. With an hFE of 40 it results in about 400mA max across the collector and emitter. This transistor won’t be PWMed but rather we will fully turn it on/off for long periods of time.

### Stepper motor control

We add the possibility of controlling a stepper motor as an electronic camera focuser. However only x3 2P connectors are provided in the board as the A4988 stepper driver wouldn’t fit. Hence this optional module should be mounted on an external board.

Its connection is straightforward following any stepper tutorial. Mainly, we need a power connection (to power the motor), a logic power connection (to power the module) and two control signals: STEP and DIRECTION. We also propose to use three jumpers to select microstepping operation. Finally, a decoupling capacitor of 100nF is added between Vmot and GND.



<https://www.ti.com/lit/ds/symlink/drv8873-q1.pdf?ts=1617255111421&ref_url=https%253A%252F%252Fwww.google.com%252F>

<https://www.ti.com/lit/an/slua979a/slua979a.pdf?ts=1596847767633&ref_url=https%253A%252F%252Fwww.google.com%252F>