University of Colorado
Anschutz Medical Campus

Optogenetics and Neural Engineering

Arduino Simple 8 Channel MEGA TTL Pulse Generator and Controller

Developed and described by Andrew Scallon
Optogenetics and Neural Engineering Core
Abigail Person and Gidon Felsen, Directors
Department of Physiology and Biophysics
School of Medicine, University of Colorado

tl;dr

A previous Arduino controlled TTL pulse generator and controller has been discussed. This paper expands on this example to allow for 8 independently controlled TTL signals. Updated housing is provided to allow for Arduino Mega use and 8 BNC connectors. An example is given for closed loop optogenetics that allows for 4 arm maze monitoring with IR Break Beams and control of ChR2 activation, Jaws activation, and a reward port.

Introduction

This document expands on previous work done with an Arduino to generate TTL pulses (see optogenetic.ucdenver.edu for more information). The previous documents showed how to fabricate an inexpensive TTL pulse generator and controller including the equipment necessary, assembly methods, software, and timing characteristics. Two examples were given to highlight how to obtain the required TTL pulse with little to no previous programming experience. The first example shows 4 independent TTL pulses delivered ad infimum. The second example demonstrated how monitor an area with an IR break beam sensor and output a TTL pulse for laser stimulation. This example expands on these for more BNC TTL inputs/outputs. See previous examples for more information on assembly and programing. All files are embedded in this document, at the bottom.

This example outlines how to set up the system for monitoring 4 arms of a behavioral maze. When the animal is detected in Arms 1 or 3, a TTL is output appropriate for ChR2 activation. When in Arm 2, there is output for Jaws activation. When in Arm 4, a reward port is opened for 4 seconds.

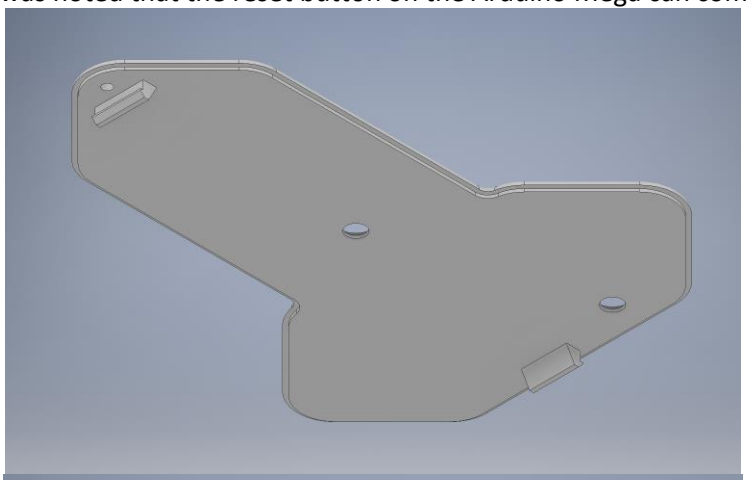
Equipment

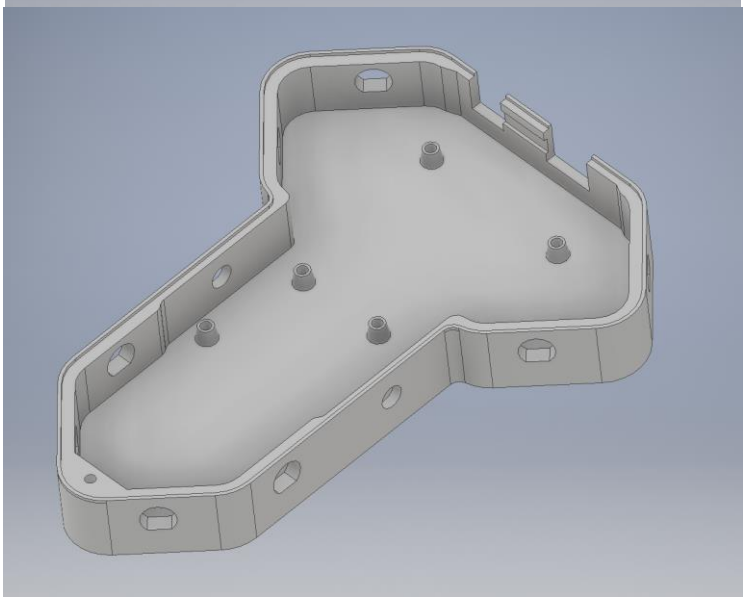
All equipment used is the same as previous examples except that the Arduino has been upgraded to the Arduino Mega (Adafruit PN 191, $45.95). The Arduino Mega inputs/outputs TTL (PWM) on pins 2-13 and 44-46.

The 3D Printed Housing

The 3D Printed Housing has been expanded to accommodate the larger Arduino Mega. It also allows for two additional protrusions for the additional BNC connectors while minimizing print material. The 3D models are given in the deisgn file format (.ipt) as well as the industry standard for 3D printing (.stl). The parts were successfully printed with nGen at 0.25 mm layer height, 1 mm shell, and 20% infill,

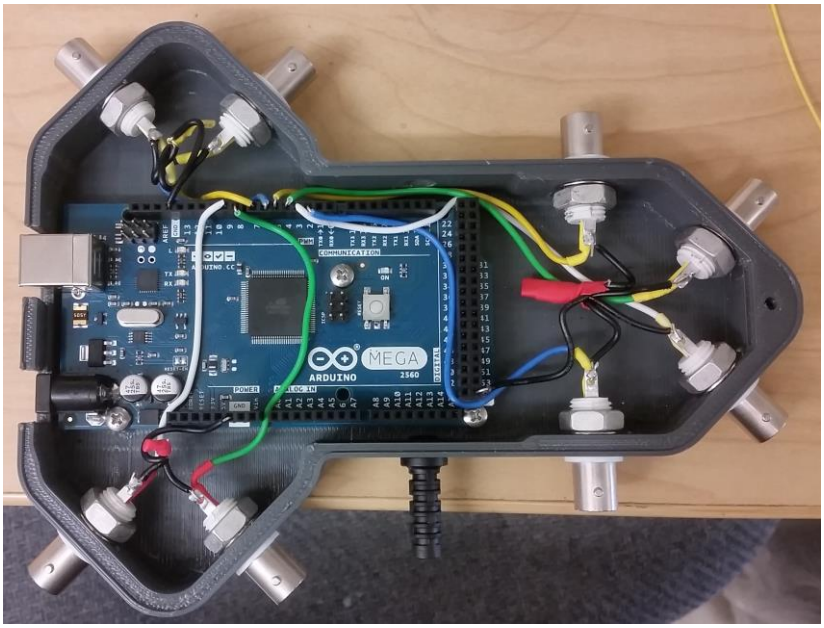
no supports needed. The top snaps to the bottom, but a M3 screw can be used as well. Two holes are put into the top as it was noted that the reset button on the Arduino Mega can come in two locations.





Assembly

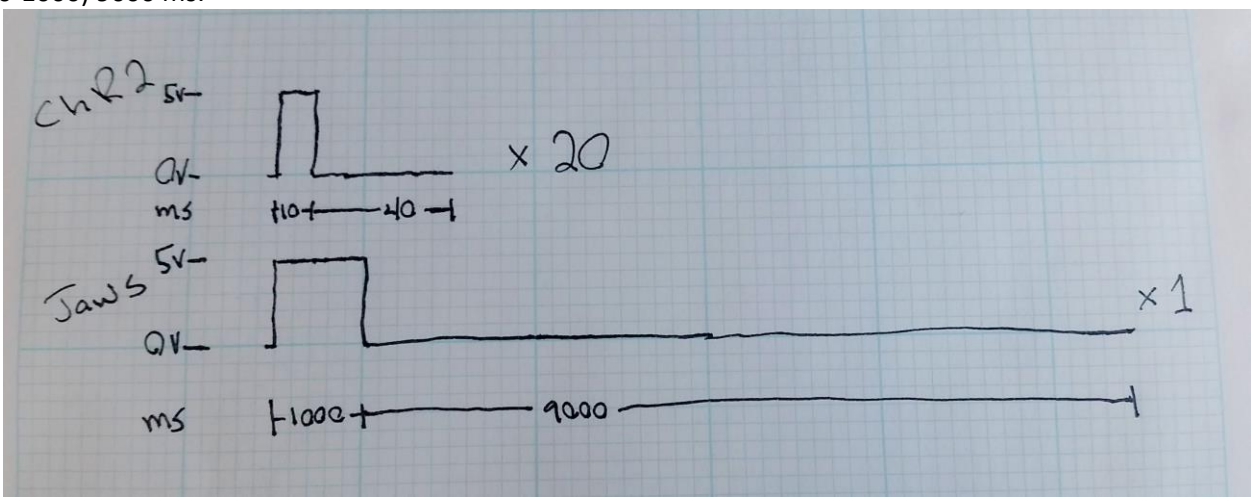
1. The assembly process is similar to the previous examples. Grounds can be soldered together to pins of nearby BNCs. Grounds can then go to available spots on the Arduino Mega.
2. In the picture below, the bottom right BNC was chosen as TTL1. It therefore routes to pin 2 on the Arduino Mega. The remaining pins are chosen to increase from there in counter-clockwise fashion, and therefore go from pins 3 to 9. It is recommended to label the TTL's directly on the housing.



Example 3: Arduino Mega 8 TTL Pulse controller

To start, **trace out the desired pulsing on paper**, with each trace on the same timeline. In this example, from literature, we note that appropriate stimulation of CHR2 is 10 ms pulses, repeated 20 times at 20 Hz. Similarly, to previous examples, it is recommended that this be drawn out. Note that Arduino works in milliseconds, so let's work in that. First draw out a 10 ms on pulse. Then note that 20 Hz is (=(1/20)*1000) 50 ms. So the pulse must be off (=50-10) 40 ms. This is repeated 20 times.

Appropriate stimulation of Jaws is 1 second pulse, repeated once at .1 Hz. So draw an on pulse for (=1*1000) 1000 ms. This is repeated at .1 Hz, or (=.1/1000) 10,000 ms. The off pulse is therefore (=10,000-1000) 9000 ms.



Again, it is recommended that the previous examples be understood before working through this example. Open the Arduino_Simple_Closed_Loop_Mega.ino file with the Arduino IDE. It is broken into sections. The first section is an introduction. The next section defines what pins we will be connecting to on the Arduino. The PWM (TTL) pins on the Arduino Mega run from 2-13 and 44-46. We will use 8 pins to control the BNC's. We will therefore use pins 2-9. We then run a set the initial state of

the sensor variable (sensorState). The Arduino will look at a sensor and determine the state (IR beam broken or not) and store the value here. These first sections will typically remain unchanged.

The next section begins to deal with the desired pulses. Here we define the duration of each condition. And the number of times the waveform is repeated.

The next section is the defining of a placeholder variable used internally by the Arduino.

The next section defines the input vs output of the TTL pulses. We have chosen the arms of the maze (1 through 4) to be represented by TTL pulses 1 through 4. TTL 5 will power the IR Beam and Sensors. TTL 6 will be the output for ChR2, TTL 7 will output for Jaws, and TTL 8 will be the reward port.

We then define the initial states of the pins. All sensors should be set to high initially. We set TTL5 high and do not change this with the rest of the code. This will deliver the power to the sensors and IR Beams. The initial state of the lasers is off. The initial state of the reward port was chosen as high (reward port closed).

The final section is the heart of the code. This is the looping code that the Arduino will loop/repeat through. This code is the same as what was discussed with Example 2, repeated 4 times (once for each IR Beam). The Arduino reads each of the sensor TTLs. If the beam is broken, it goes through the indented code. For example, it first reads TTL1. If TTL1 is low (IR Beam is broken), it executes the indented code. That is, for TTL1, it cycles through *for* the number of times the waveform is repeated (for ChR2 we chose 20), writing the TTL high (laser on), waiting for the number of ms the waveform is on (we chose 10), writing the TTL low (laser off), waiting the number of ms the waveform is off (we chose 40), then repeating until we reach the number of times we said to cycle through (we chose 20). When done, reset to the sensor to the high state (as was done initially). We then reread the state of the sensor to ensure that the stimulus is delivered only once while the animal breaks the IR beam. We wait 500 ms (.5 seconds), and then reread the sensor. This repeats *while* the sensor is still broken. The Arduino then moves on to the next IR sensor.

Supplementary Files

Files may be opened directly from this Word Document, or maybe copied into a folder of your choice.

Screenshots of Downloading and Pro

Screenshots of installing Arduino Software and uploading programs to the Arduino.

Arduino_Simple_Closed_Loop_Mega.ino

The Arduino code for Example 3: Simple 4 Channel TTL Pulse Generator

Arduino simple 4 channel ttl bottom M

The Design File (for modification) of the 3D housing bottom

Arduino simple 4 channel ttl top Mega.

The Design File (for modification) of the 3D housing top

Arduino simple 4 channel ttl bottom M

The 3D Model for printing the housing bottom

Arduino simple 4 channel ttl top Mega.

The 3D Model for printing the housing top

Please acknowledge our facility in your publications. An appropriate wording would be:

"Engineering support was provided by the Optogenetics and Neural Engineering Core at the University of Colorado Anschutz Medical Campus, funded in part by the *National Institute for Neurological Disorders and Stroke* of the National Institutes of Health under award number P30NS048154."