OpenChrony V1.0-1.5 Build Guide

Here I present an open source chronograph, compatible with nerf and paintball. It is Arduino-based, and the code is publicly available, as are the part files. I’m hoping that others will improve the design, this is only the first functional version.

Code and Models

Thingiverse link to 3D-printable file set and Arduino code

Bill of Materials

Arduino Uno, I used this kit which contains other parts in the BOM

- 1x

Protoboard, 4cm x 6cm

- 1x

LM324N Op Amp

- 1x

Jumper Wires

IR LEDs & Photodiodes (can likely be had for cheaper on Mouser)

- 2x IR LED’s
- 2x Photodiodes

Various LED’s (come with Uno kit)

- 2x Blue (V1.0 only)
- 1x Green (V1.0 & V1.5)
- 1x Yellow (V1.0 only)
- 1x Red (V1.0 only)
Various Resistors (come with Uno kit)
- 2x 150 Ohm (depends on specs of specific IR LED’s used)
- 5x 330 Ohm
- 2x 1 MOhm

**Adafruit Seven Segment Display**
- 1x

**10 kOhm Potentiometers** (can likely be had for cheaper on Mouser)
- 2x

**Pin Headers**
- 1x row of 2 pins

**Slide Switch** (can be had cheaper on Mouser)
- 1x

**12mm Push Button NO Momentary Switch** (cheaper elsewhere, just get 12mm version for compatibility with file set)
- 1x (V1.0 only)

M3x10mm Phillips-Head Screws
- 9x

M2x10mm Phillips-Head Screws
- 10x

**Instructions**

**Important note for V1.5 firmware:**

The following components have been removed and obviously don’t need to be installed. Ignore the associated bits of the write-up.

- **ALL** LED’s besides the green
- Momentary push-button

Print all parts from the Thingiverse link. Nothing is going to undergo much stress so do whatever settings you feel comfortable with. Grab the IR_Tunnel_split_1 and IR_Tunnel-split_2 and super glue them together. Depending on your printer accuracy, you may have to shave some plastic off the male alignment pins.

Bolt the Uno into Electronics_Box_Bottom with 4x M2’s.
Bolt the IR tunnel to Electronics_Box_Bottom with 2 M3’s, but only thread the screws halfway into their respective holes. These will have to be removed at the end to bolt Wire_Sheath in place, but it’s easiest to have assembled during the wiring stage of the build.

Circuit for one of the Op-amps:

The diode with arrows going into it is the photodiode. Note that it is wired in reversed bias, i.e. the long leg goes to ground instead of 5v in this case. The 10k resistor with an arrow to the middle is the 10k pot trimmer.

Op-amp pinouts:
Image courtesy of Wikimedia Commons. Only OPV_A and OPV_D are used here, and obviously pins 4 and 11 for power/ground.

It is highly recommended to lay out one of the IR gate circuits on a breadboard before soldering anything in place. It helps a lot to understand how the circuit works beforehand.

Circuit soldered to protoboard:

Important note: I miscounted how many ground pins I needed. Part way through assembly I needed to solder grounds together. Don’t do that! I have 7 ground pins pictured in that clump on the top right, minus those used up by the photodiodes and pot trimmers. I believe I was 2 pins short. Make 10 dedicated ground pins, just to be safe.

Here is the reverse side:
Notice how one of the outputs is orange, and the other is brown? Doing something like this is highly recommended. Since this is essentially two identical circuits on one board, it is easier to keep track of which is which if with color-coding. In this image, orange is the first gate, and brown is the second.

Ever so carefully test out the circuit. Those wires coming off the board are fragile and will break at the solder joint very easily. If it doesn’t work, check all of your connections with a multimeter. Make sure nothing is shorted that shouldn’t be, and that all your solder joints have 0 Ohms of resistance.

At this point, those four joints need to be strengthened where the jumpers meet the protoboard. Hot glue is perfect for this.
Wire up the LED’s to resistors and jumpers. The IR LED’s presented here (clear) only have 150 Ohm resistors because of their specs. The colored (non-IR) LED’s use 330 Ohm resistors.

Note that in the image, the colors on the positive end are consistent with the colors of the LED’s. Except the yellow and one of the blues, because of a mistake.

Wire up the pot trimmers. Remember the color coding on the op-amp outputs? Do that again here, as each pot is associated with one of the gates.

Wire the two-position slide switch however you see fit. I used male headers. Just keep track of the jumper colors used later.
Wire up the push-button next. Ground should be black and the color for the Arduino side doesn’t matter, just keep track of it.

Last up is the photodiodes. Note that this build needed other wire to extend the jumpers, as single jumpers wouldn’t have been long enough. While not presented here, it is recommended to follow the established color-coding scheme for each gate.

Time to decide where all the I/O wires to the Arduino will go.
Note that 2 and 3 need to stay as they are (for future versions of source code that utilize external pin interrupts), but 4-12 can be wherever. A4 and A5 also need to stay as they are, as the Adafruit display requires them.

Start by plugging all I/O wires into the Arduino.

<table>
<thead>
<tr>
<th>Arduino I/O pins</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 – gate 1 opamp out (orange)</td>
<td>(for future versions of source code that utilize external pin interrupts)</td>
</tr>
<tr>
<td>3 – gate 2 opamp out (brown)</td>
<td></td>
</tr>
<tr>
<td>6 – green LED positive (green wire)</td>
<td></td>
</tr>
<tr>
<td>7 – yellow LED positive (blue = D in dumb)</td>
<td></td>
</tr>
<tr>
<td>8 – red LED positive (red)</td>
<td></td>
</tr>
<tr>
<td>9 – gate 1 blue LED (a) (blue)</td>
<td>(Actually useful that it’s dim)</td>
</tr>
<tr>
<td>10 – gate 2 blue LED (b) (yellow)</td>
<td></td>
</tr>
<tr>
<td>11 – reset button (red)</td>
<td></td>
</tr>
<tr>
<td>12 – mode toggle (purple)</td>
<td></td>
</tr>
<tr>
<td>A4 – “D” pin on LED display</td>
<td></td>
</tr>
<tr>
<td>A5 – “C” pin on LED display</td>
<td></td>
</tr>
<tr>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>Gate 1 pot → orange</td>
<td></td>
</tr>
<tr>
<td>Gate 2 pot → brown</td>
<td></td>
</tr>
</tbody>
</table>

Bolt on the protoboard with 2x M2’s.

Grab Electronics_Box_Top. Insert the trimmers and drip hot glue on the back to keep them in place.
Start gluing/bolting the rest of the components into Electronics_Box_Top. The Adafruit display attaches with 4x M2’s. For the LED’s, put a bead of hot glue on the side, close to the base, and then insert them into their respective holes. Below is everything attached to Electronics_Box_Top.

Install the IR LED’s and photodiodes in the IR tunnel.

Wire up the protoboard.
Carefully close everything up. This will take some force, but you need to make sure wires aren’t getting pinched by the plastic housing or anything like that. To get the inner bolts for the lid on, detach the IR tunnel. Electronics_Box_Top onto Electronics_Box_Bottom with 4x M3’s.

After Electronics_Box_Top is bolted in place, bolt Wire_Sheath to Electronics_Box_Top and then re-attach the IR tunnel. The four wires from each gate can be popped into Wire_Sheath one-by-one, even with the inline resistors. Then simply bolt on Wire_Sheath with 3 more M3’s.

The build is complete. Here is a video showcasing operation.
Future Work – Version 1.x Intended Revisions

Pin interrupts instead of do-nothing while loops to make sure micros() gets called at exactly the right time. This would allow the chronograph to continuously be in “shooting mode”, and “troubleshoot mode” could be entered at any time. Basically, there would be no “red mode” when the red LED goes off. The FPS from the previous shot would just stay on-screen until the next reading. This would also prevent the need to manually clear the rear gate if the dart misses.

Mass reduction on printed parts. Wall thicknesses are 10 mm at a minimum, except for the electronics box top. This is wildly unnecessary. Also, more material can be carved out of the IR tunnel, specifically under the tunnel itself. The parts were designed this way selfishly; I personally could care less about filament consumption and like the bulkier appearance, even if it isn’t functional.

Incorporate adapters for standard paintball or nerf barrels. Barrel alignment with the gates would then be a non-issue and projectiles would almost always clear both gates.

Write something better than the native micros() function to record timing. As it is right now, this setup gets around 12 microseconds of inaccuracy max. At 200 fps this is around +/- 2 fps which is acceptable for the purposed of this chronograph but can be improved upon.

Future Work – Plans for Version 2.0

Down-size the chronograph for direct mounting to nerf blaster/paintball marker muzzle. The Uno will be swapped out for a microcontroller with a smaller form factor, likely an Arduino Mini. The entire chronograph - gate, electronics, and all - will be contained in a cylinder that is affixed to the muzzle. Output LED’s will be eliminated by better utilization of the seven-segment display. The 12mm pushbutton will be eliminated with smarter code (see first intended revision to 1.x).

Custom, printed PCB. I’m currently teaching myself EagleCAD so I can design a PCB and order it from a to-be-selected manufacturer. This would eliminate the need for the complicated protoboard circuit and components would simply need to be soldered in place. These could potentially be for sale in the future.

Eliminate tethered design. Instead of needing a power cord, the chronograph will be battery-powered.