Automatic Breadboard Wire Cutter v1.0

Developer Guide

A picture containing indoor, wall, floor, sitting

Description automatically generated

ITP 348 Final Project

Ben Boggs – boggsb@usc.edu

1. Hardware

Device Schematics:

A close up of a device

Description automatically generated

A picture containing indoor, wall

Description automatically generated

Complete CAD was drawn in SolidWorks, which is available in the SAL computer lab.

A full parts list and procurement method is shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Part | CAD file (same as Part if blank\*) | Procurement Method | Price (est.) |
| Base |  | Laser Cut | 1 12”x24” 1/8” thick plywood sheet ~ $12 |
| Spool |  | 3d Printer | - |
| Spool Support |  | 3d Printer | - |
| Wire Table |  | 3d Printer | - |
| Blade Support |  | 3d Printer | - |
| Blade Guard |  | 3d Printer | - |
| Blade Lock |  | 3d Printer | - |
| Wire Guide |  | 3d Printer | - |
| Steel Wire ~1mm | - | Owned | $0.50 |
| Brass tube .91” ID | SS tubing .071 ID | Owned | $4.00 |
| Xacto blade #11 | Xacto blade – No 11 | Owned | $1.00 |
| TowerPro MG92b Servo | SERVO MG92B | Bought | $14.95 |
| 4x M3x0.5 16mm Socket Cap Screws | M3x0.5 L14 | Bought | $3.08 |
| 3x M3x0.5 20mm Socket Cap Screws | M3x0.5 L18 | Bought | $2.49 |
| 7x M3 Hex Nuts | M3x0.5 Hex Nut | Bought | $2.17 |
| Sandpaper | - | Owned | $0.10 |
| Misc. Fastening HW | - | Owned | $0.50 |
| 2x Breadboards | breadboard | Course Kit | $9.90 |
| 2x Motors with fasteners | DC motor with gearbox | Course Kit | $4.95 |
| 2x Wheels with fasteners | Wheel | Course Kit | $4.00 |
| Magnetic Door Switch | Door Reed Switch | Course Kit | $3.50 |
| Sparkfun Motor Driver | - | Course Kit | $5.45 |
| Phototransistor | - | Course Kit | $0.95 |
| 2x 10 kΩ Resistor | - | Course Kit | $0.12 |
| 330 Ω Resistor | - | Course Kit | $0.05 |
| Jumper Wires | - | Course Kit | $2.25 |
| Particle Argon | - | Course Kit | $27.00 |
| TOTAL\*\* |  |  | $100 |

\*Some parts are defined in assembly context only. To access, open the Wire Cutter Assembly file, and right click its name on the left and select “Open Part”.

\*\*Does not include Tax, Shipping, or 3D Printer Filament costs

The full assembly CAD file is named Wire Cutter Assembly and can be used as a guide for assembly. The phototransistor was set into the hole of the Spool, and wires fed through the slot on the side to avoid interfering with wires. The Base was made by laser cutting each of the 4 configuration “layers” out of 1/8” thick plywood, then laminating them together in order. Configurations can be viewed in the Base SolidWorks file. It is a tight fit with all the wires under the Wire Table, so as many properly cut wires (ie, flat, not the jumpers) were used as possible to get the table to fit. Additionally, the wheels were wrapped with sandpaper to aid in gripping the wires.

A wiring diagram was developed in fritzing, shown below:

A circuit board

Description automatically generated

Known Issues:

* The motors are not strong enough to extrude the wires properly. Switching to a stronger motor would resolve this, but motors also must be spaced carefully to tune the force on the wire
* The motor travel is inconsistent. This could be fixed in tandem with the previous problem by using a stronger stepper motor instead of a geared dc motor.

1. Software

The software developed for this project consists of the Argon firmware (C++), python support code (python), and Losant interface app (javascript).

No external libraries are required to run any code. Refer to the User Documentation on how to use the python script and Losant App.

Argon Firmware

The code is organized into 5 labeled sections:

* *Declarations*: function headers, and underlying struct definition.
* *Pins*: Pin definitions. All following code refers to these names, so change the pin assignments to your setup.
* *Constants*: These are constant values that tune the functionality based on hardware differences, and the state enumerations. Treat these like public, global variables.
* *Variables*: These are underlying variables for core functionality. Treat these like private variables.
* *Functions*: code implementation.

The logical structure is of a finite state machine, with the states outlined in the User Documentation. States are executed by the switch block in loop(). Whenever the state is changed, the state is published to the cloud app. The state machine logic can be described with the diagram below:

A picture containing text

Description automatically generated

Jobs are represented in code as a sequence of bbWire objects. Each bbWire has 2 fields as defined by the struct at the top: the length of wire to cut in mm, and a boolean that is true when it is the last wire in the job (false otherwise).

When a job is sent to the Argon, the function queueJob() is called. This function creates new instances of bbWires for each wire length and adds them to the queue. It also calls printQueue(), which publishes the queue contents to the cloud for the app.

When the queue is not empty, and there is wire stocked, the cutting state will call cut() on the next bbWire in the queue until it reaches one with end = true, signifying the end of the current job.

The code treats the table as the magnetic door switch. When the table is picked up, the “door” is open; when it is put down, the “door” is closed.

The code digitally observes the wire stock with the phototransistor. When there is something (wires) in front of the sensor, it is stocked. When there is nothing close to the sensor, it is low on stock.

Final Built Product:

A picture containing wall, bicycle, indoor, floor

Description automatically generated