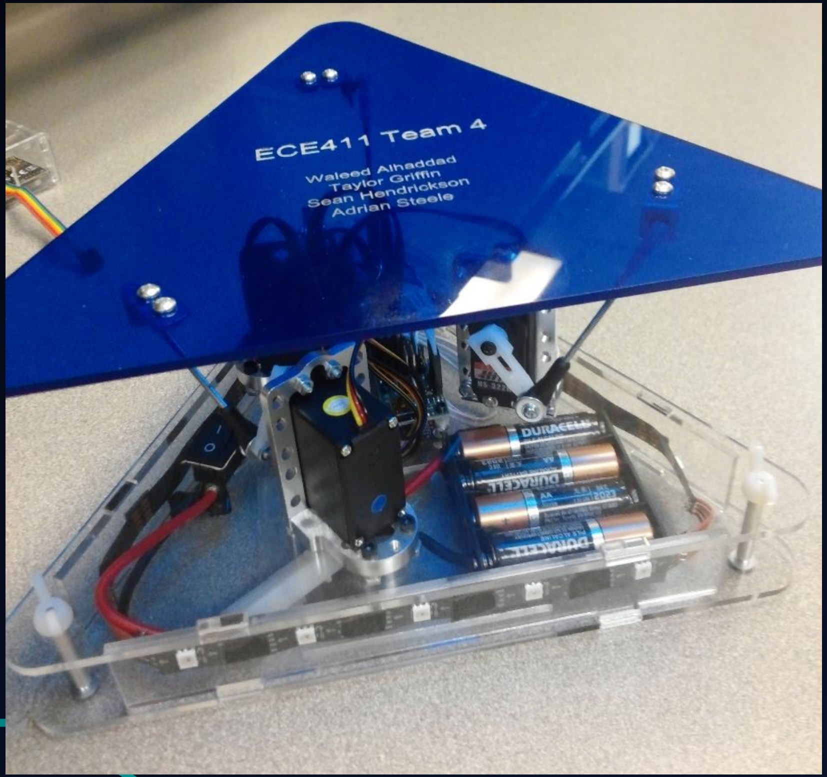
Team 04 – Self Leveling Table

SEAN
WALEED
ADRIAN
TAYLOR



Problem or Need

- Suitable project for practicum.
- Waiters sometimes drop their tray’s contents.
- Hard to keep things balanced.

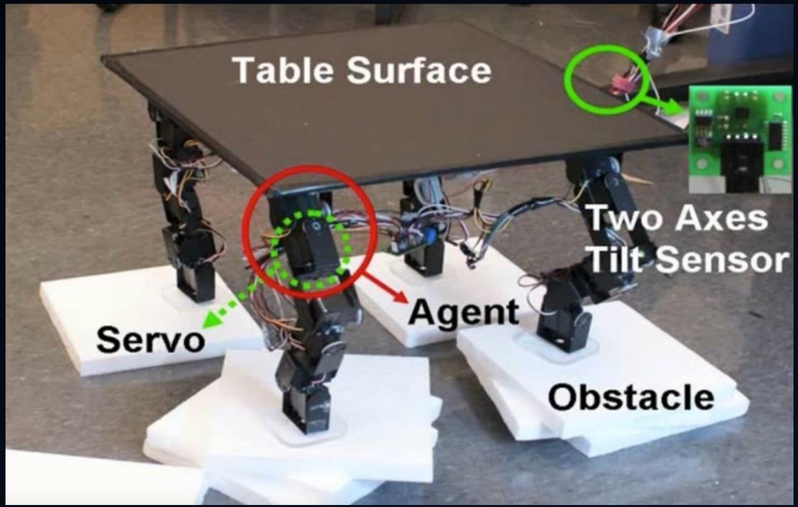
Motivation

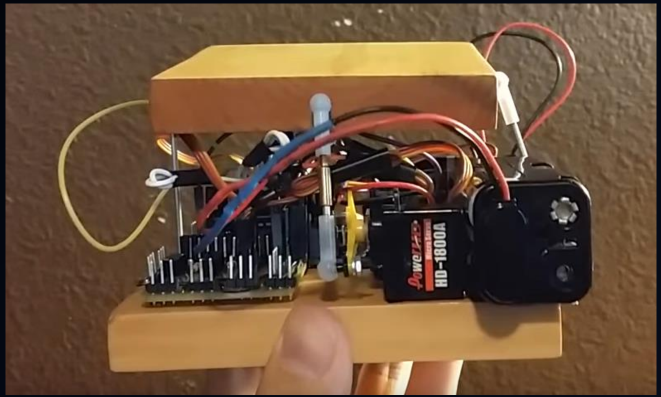
- Get a good grade!
- Resume material.
- Good practice for capstone and career.
- Cool gadget to play with.
- Useful to keep things balanced.

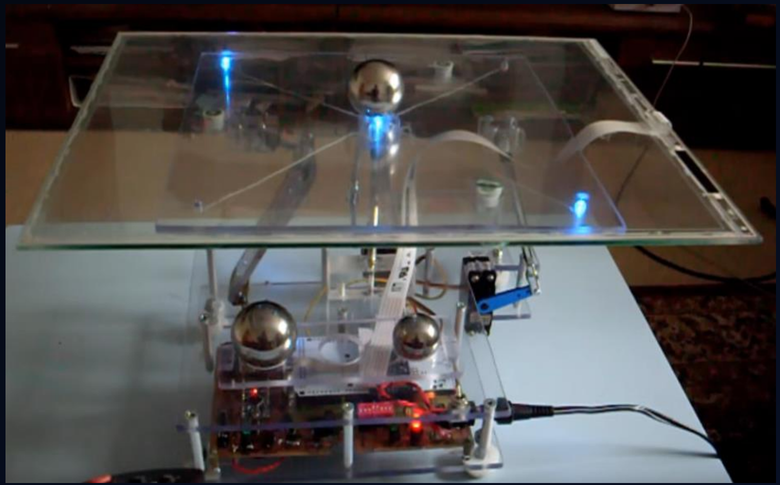
Objective

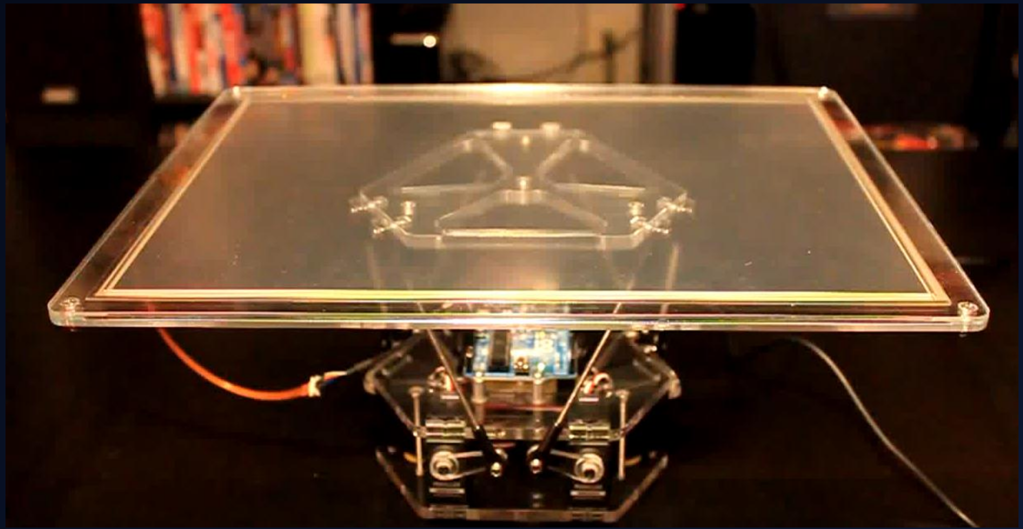
- Keep a tray level in real time.
- Create working prototype in 10 weeks.
- Entertaining to play with.

Alternatives









Marketing Requirements

- The device should be easily held with two hands.
- The device should hold an object on the table to balance.
- The device should be responsive.
- Should be simple and intuitive to use.
- Should be visually appealing.
- Should be inexpensive.
- Should be safe.

Constraints

- Must have at least one input (sensor) and at least one output (actuator).
- Must have at least 2 layers on PCB.
- Must have a microprocessor located on PCB.
- Must have more than 25% surface mount components (1206).
- Must have top-side silk screen.
- Must not start from existing design file.
- Must have live documentation and use revision control.
- Must be published with MIT license.

Engineering Requirements

- **Functionality**
  - The device should maintain a level position while powered on.
- **Performance**
  - The device should level itself within 5 degrees.
  - Should move to a level position within 2 seconds of becoming unleveled.
  - Table should remain level up to 15 degrees relative to the horizontal.
- **Economic**
  - Total parts cost will not exceed $200.

Engineering Requirements

- **Energy**
  - The system will should run off batteries for at least 1 hour.
- **Health & Safety**
  - Power system and components should be inaccessible to user.
- **Environmental**
  - Should be manufactured with lead free solder.
- **Manufacturability**
  - PCB should be greater than 1.5 sq. inch and smaller than 140 sq. inches.

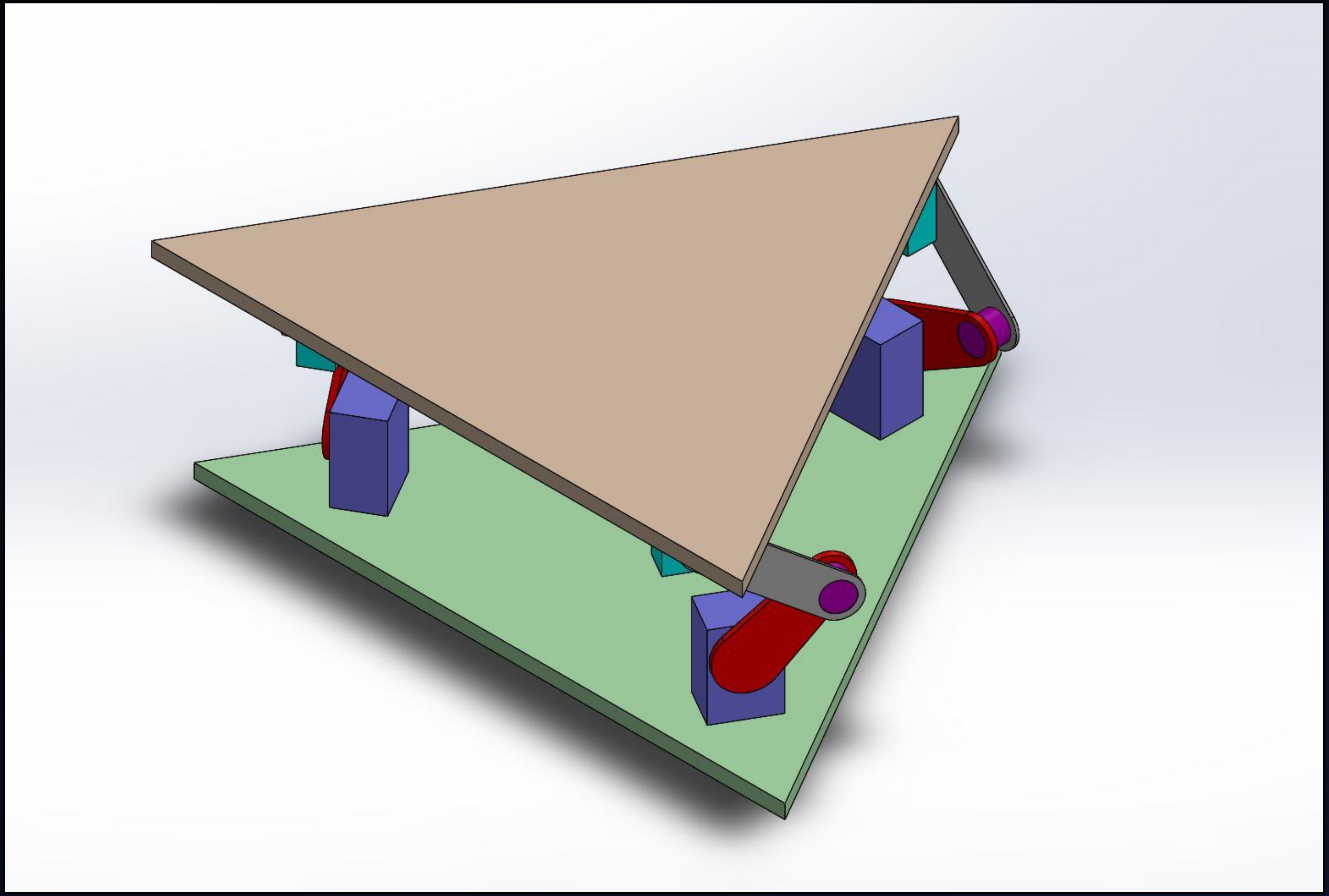
Engineering Requirements

- **Operational**
  - Device should be smaller than a cubic foot.
  - Device should not weigh more than 5 lbs.
- **Reliability & Availability**
  - Table should hold .25 lb. without breaking.
- **Social & Cultural**
  - Should have visual feedback system to relay angle state to user.
- **Usability**
  - User should not need instructions to use device.

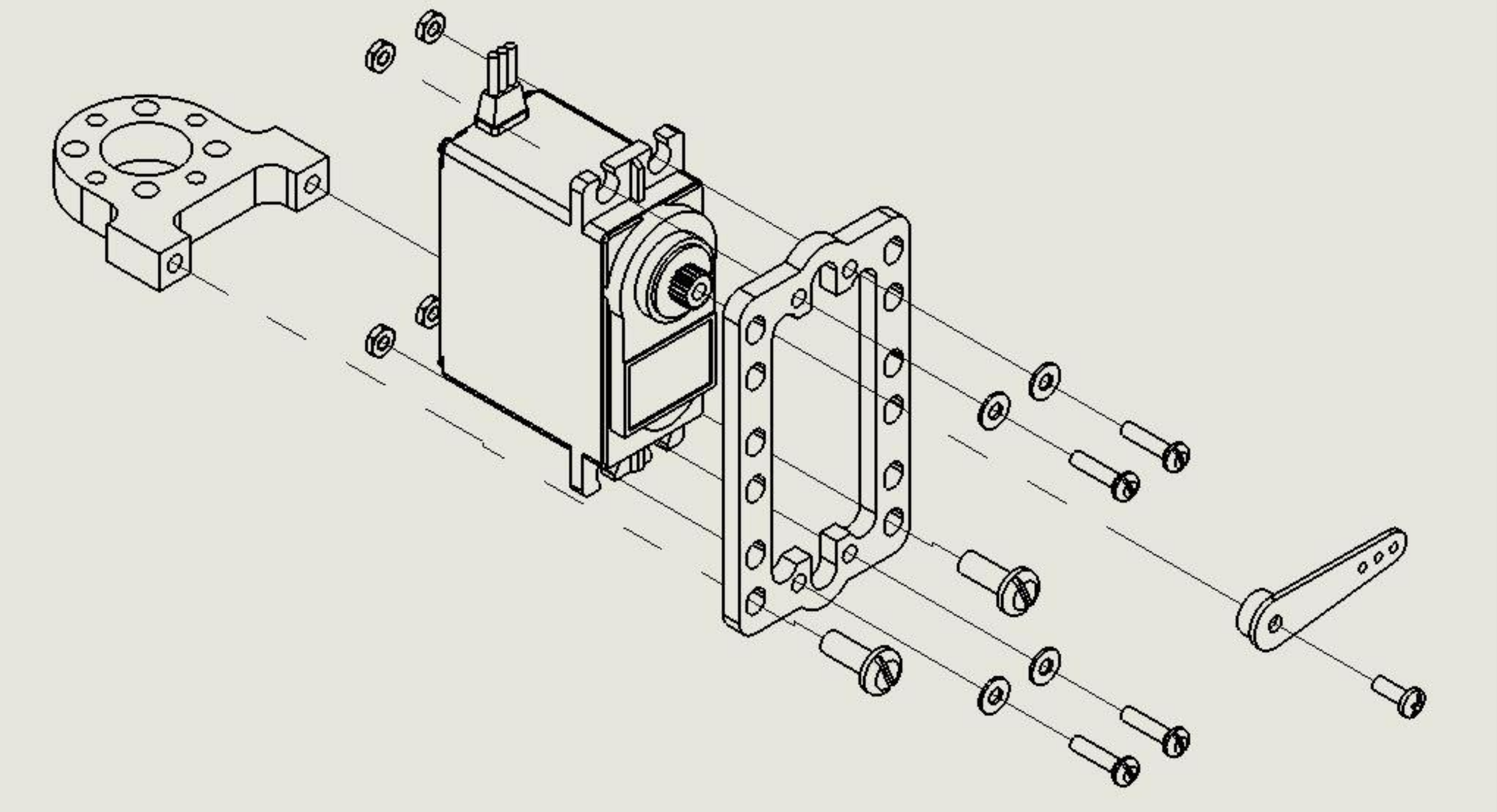
Our Approach

- Triangular base with triangular top tray.
- Three servos attached on the corners of the tray.
- Receive input from an accelerometer and process in a microprocessor.
- Send updated positions to servos based on device tilt.
- Battery powered for portability.
- Lots of LEDs for entertainment.

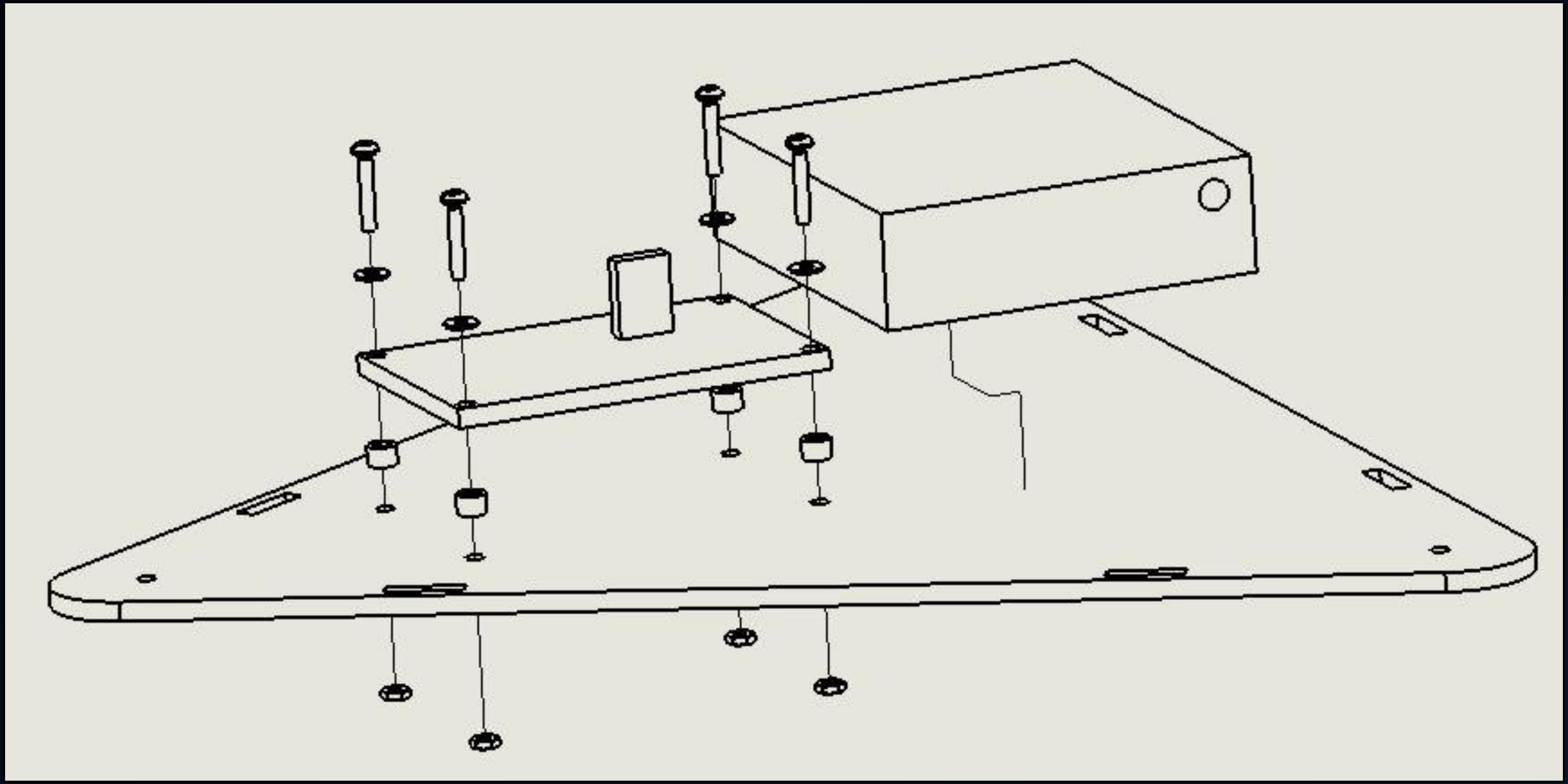
Initial Concept



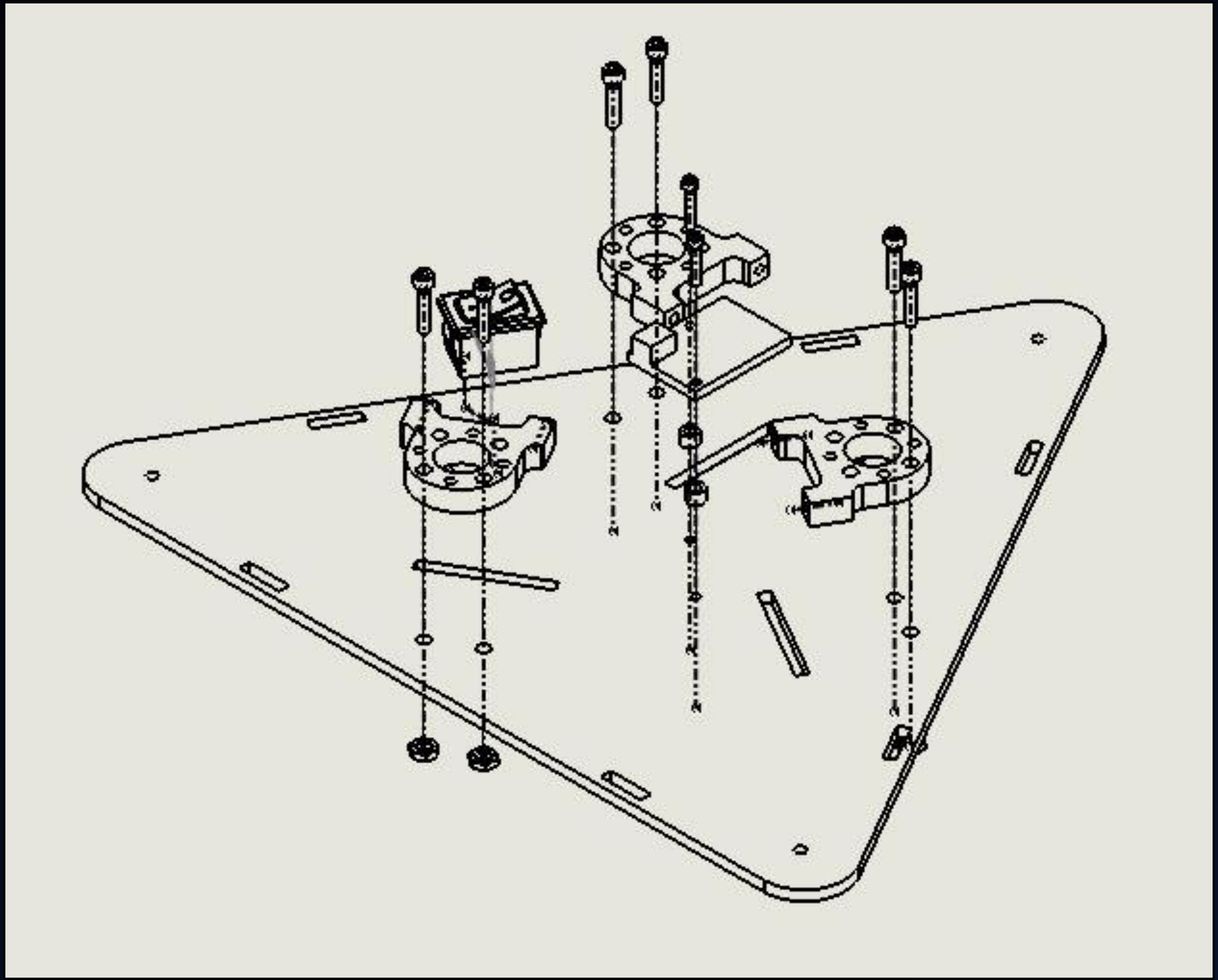
Servo Brackets



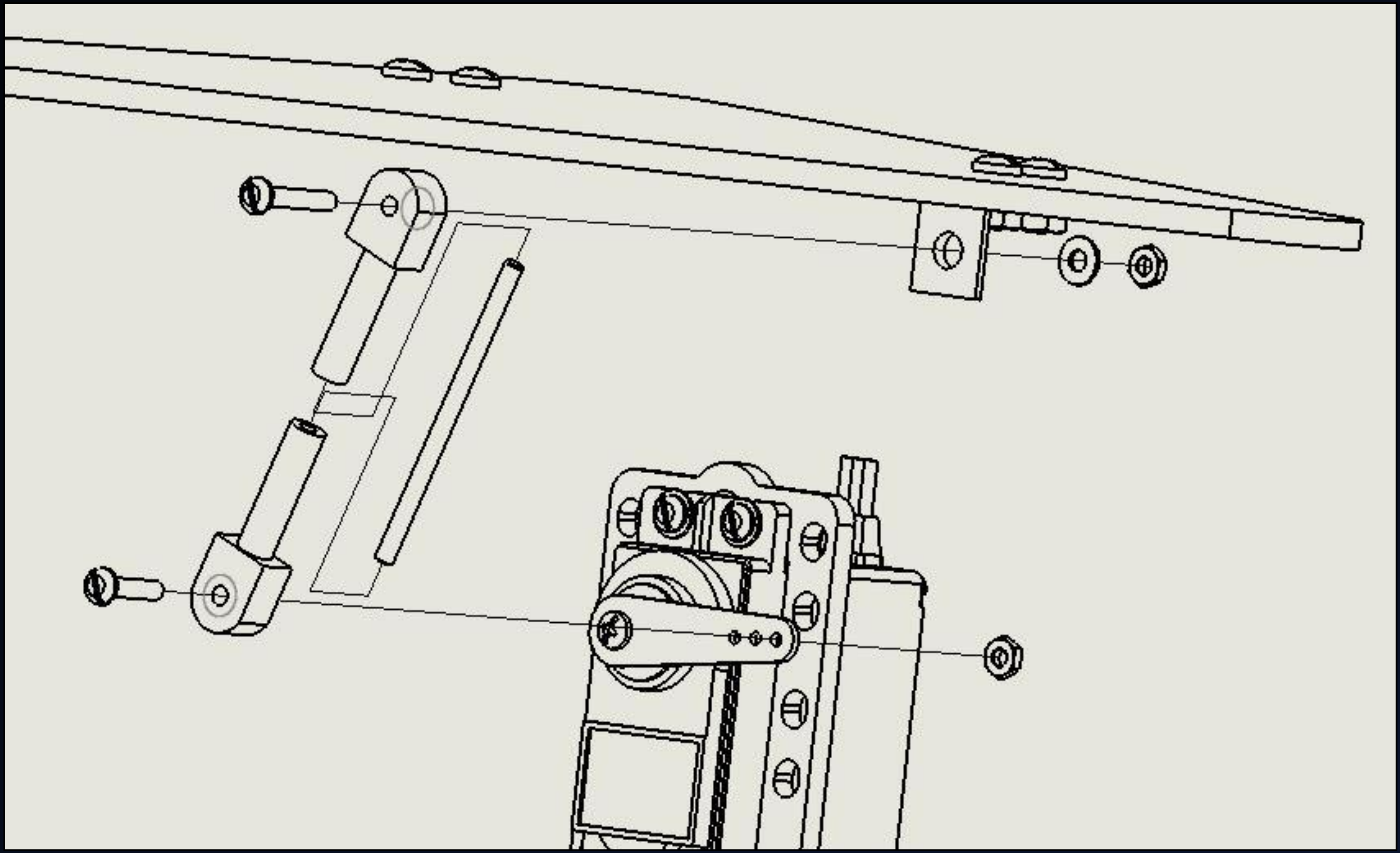
PCB & Battery Placement



Base to Servo Brackets Mounting

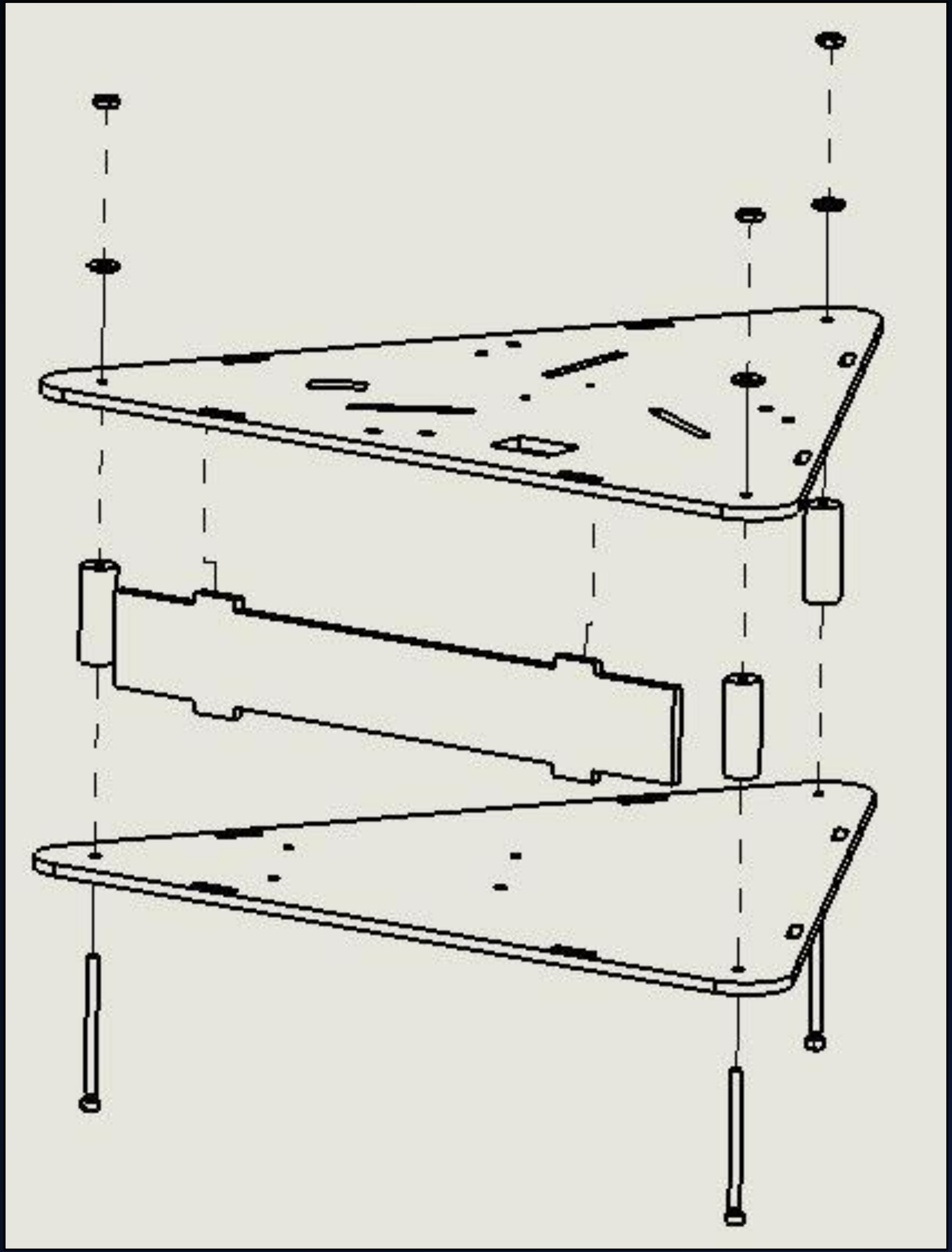


Tray to Base Assembly

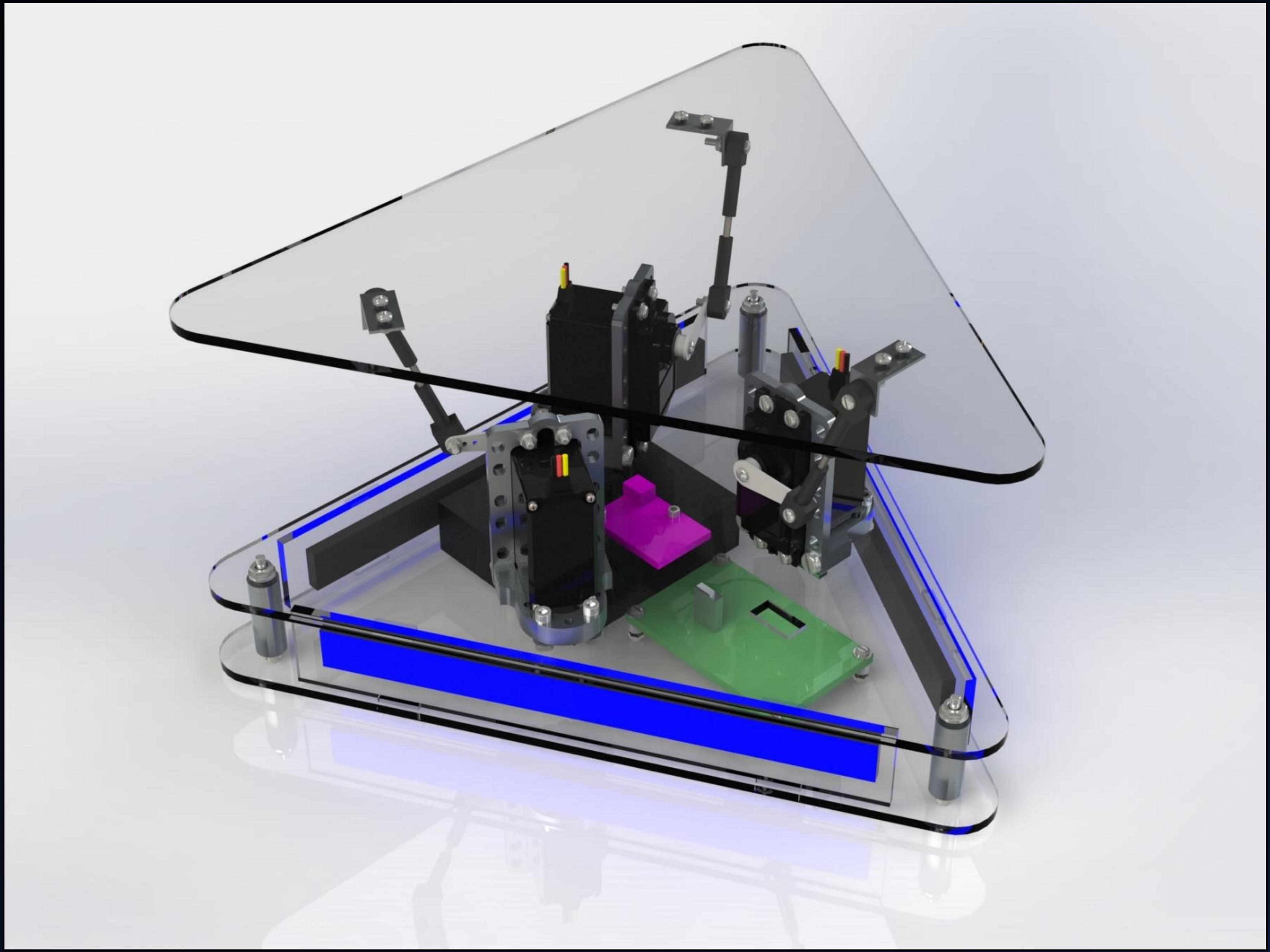


Base Assembly

- Base attached with aluminum standoffs
- Acrylic side panels used for LED mounting



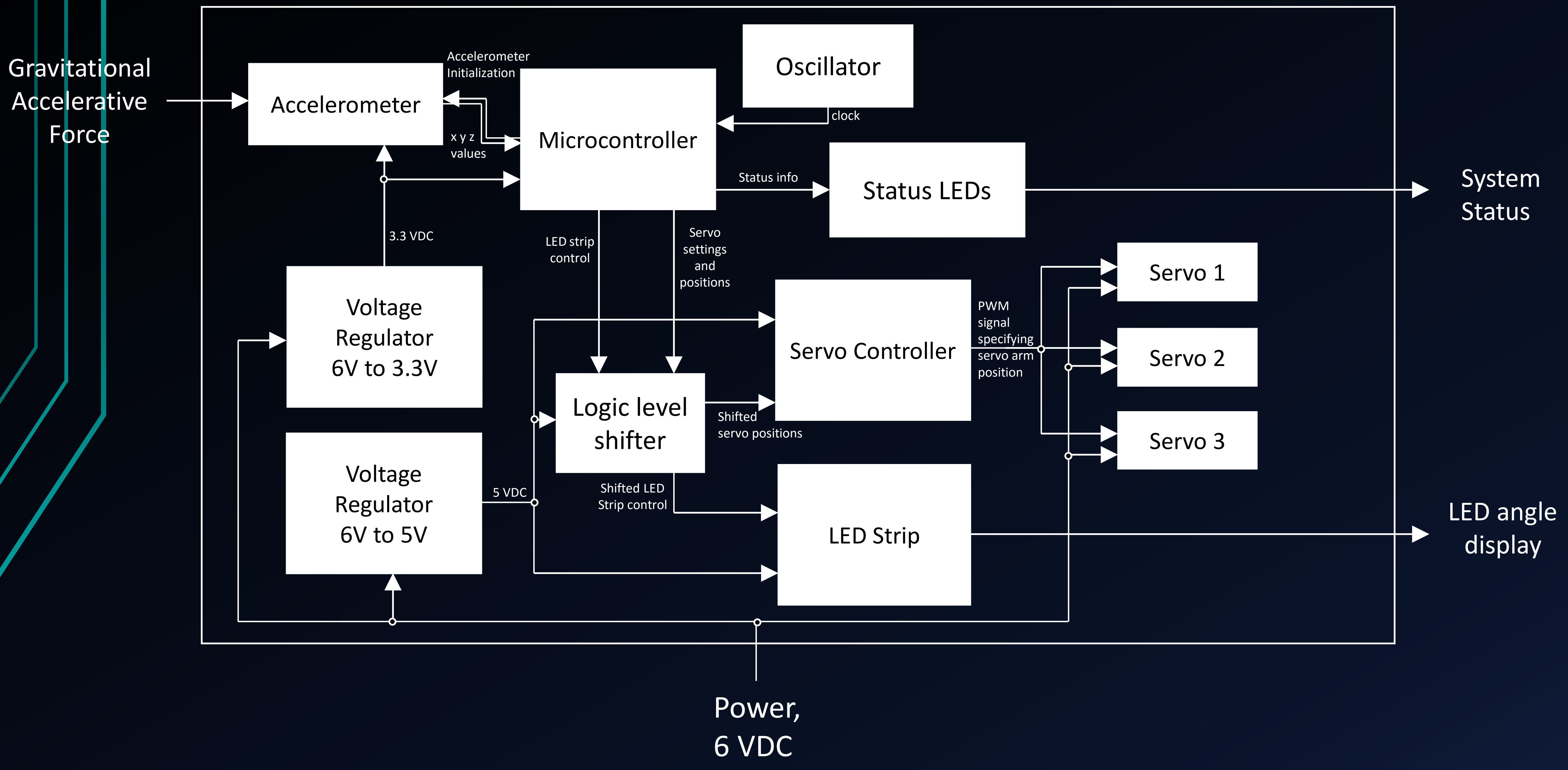
Final Design



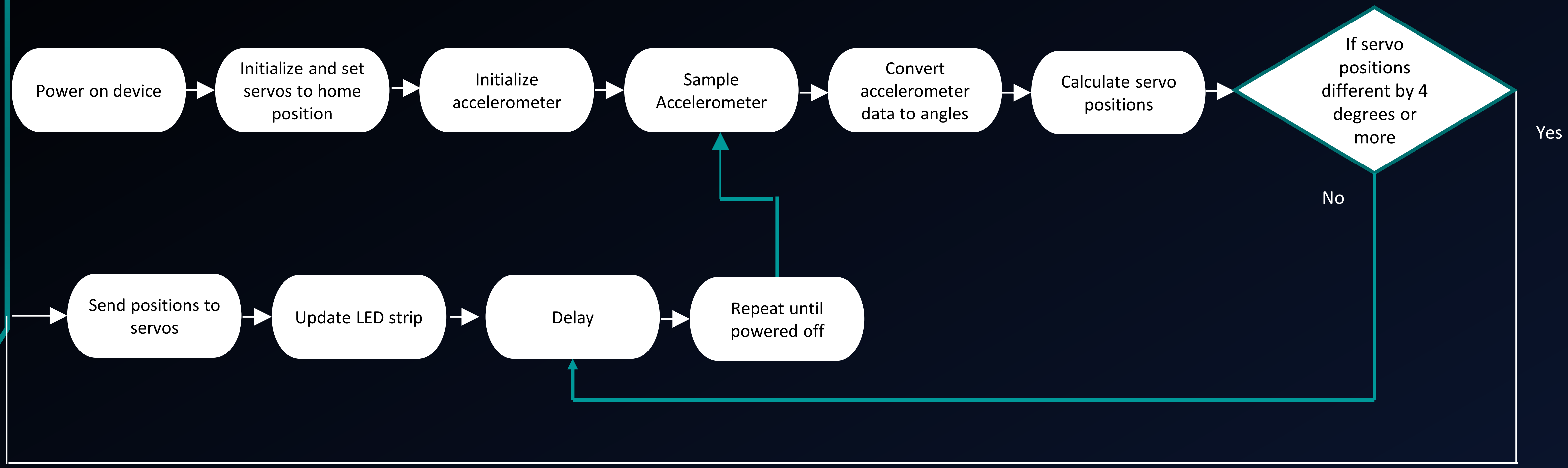
Self Leveling Table: Level 0

Module	Self Leveling Table
Inputs	- Gravitational Accelerative Force: The external forces acting on the device, used to determine the angle of the device and set servo position - Power: 6 VDC from batteries.
Outputs	- LED Angle Display: 15 LEDs lit up in different configurations. Lights up based on device tilt slowly changing colors. LEDs around the device become more blue on the device side that is tilted up. LEDs around the device become more red on the device side that is tilted down. - System status: 4 LEDs that light up on PCB. First LED is green, displays when tilt on the x axis reaches its limit. Second LED is yellow, displays when there is communication between the microcontroller and the servo controller. Third LED is yellow, displays when there is communication between the microcontroller and the accelerometer. Fourth LED is red, displays when the tilt on the y axis reaches its limit.
Functionality	- Consists of 2 layers: the base (two triangular acrylic sheets enclosing the batteries, PCB, servos, and LEDs), and the top tray (one triangular acrylic sheet leveled by servos). Receives input from an accelerometer and determines servo position to keep the top tray level.

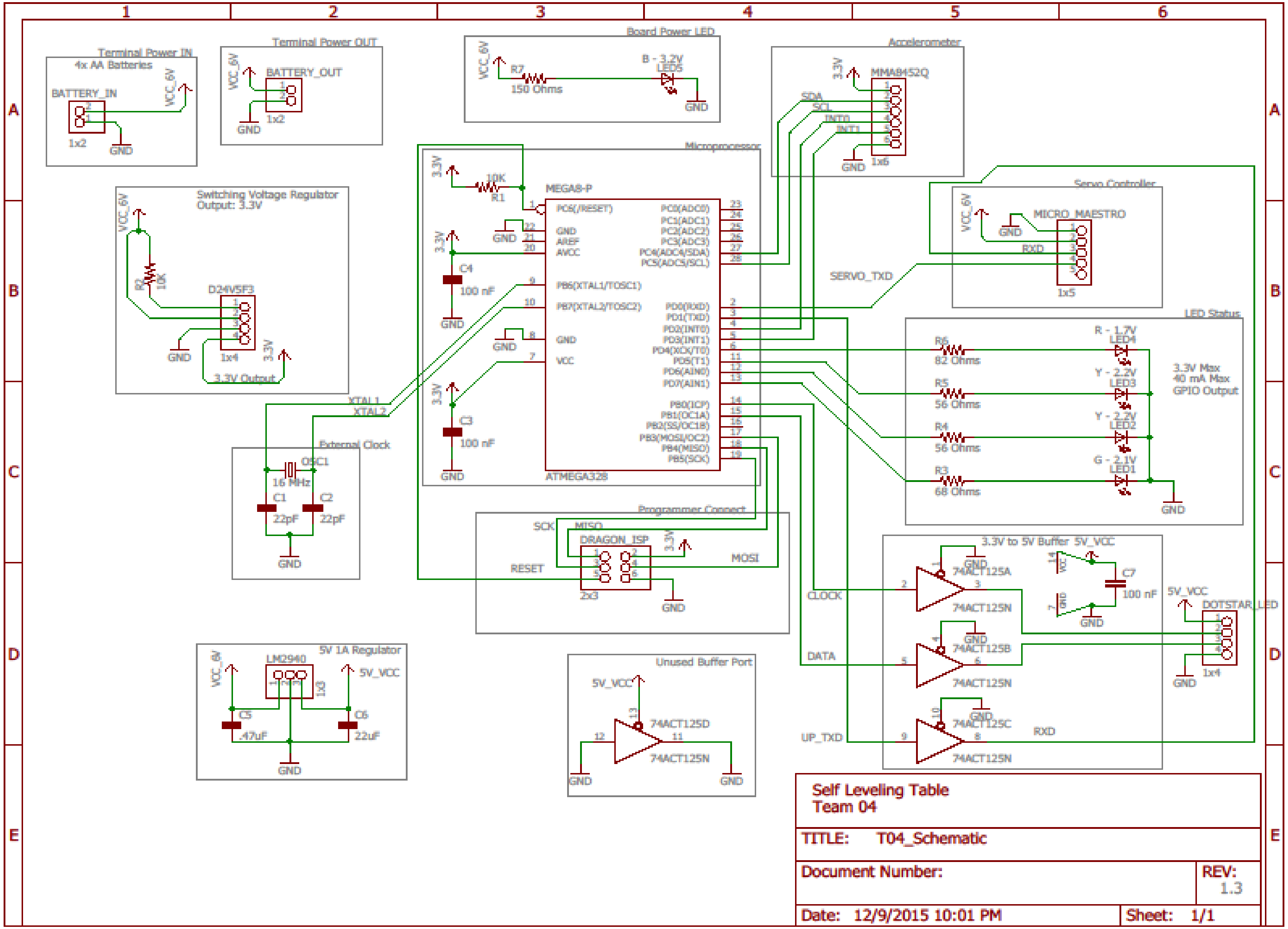
Self Leveling Table Design: Level 1



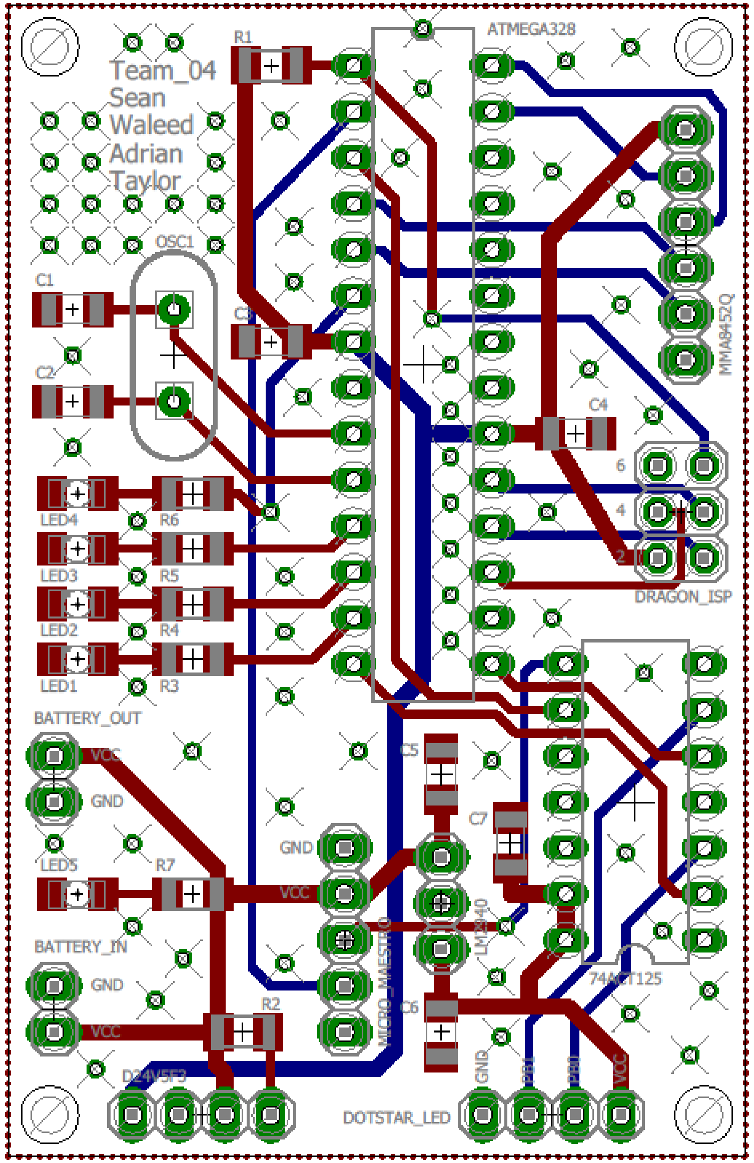
UML: Activity Diagram



*Note: Update status LEDs after every communication and calculation and in the event of an error.



Terminal Power IN
4x AA Batteries
BATTERY_IN
1x2
VCC_6V
GND
Terminal Power OUT
BATTERY_OUT
Board Power LED
R7
150 Ohms
B - 3.2V
LED5
Accelerometer
MMA8452Q
SDA
SCL
INT0
INT1
1x6
Microprocessor
MEGA8-P
10K
R1
ATMEGA328
PC6(/RESET)
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
AREF
AVCC
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
C4
100 nF
C3
100 nF
Switching Voltage Regulator
Output: 3.3V
R2
10K
D24V5F3
1x4
3.3V Output
Servo Controller
MICRO_MAESTRO
RXD
1x5
SERVO_TXD
LED Status
R6
82 Ohms
R - 1.7V
LED4
R5
56 Ohms
Y - 2.2V
LED3
R4
56 Ohms
Y - 2.2V
LED2
R3
68 Ohms
G - 2.1V
LED1
3.3V Max
40 mA Max
GPIO Output
XTAL1
XTAL2
External Clock
OSC1
16 MHz
C1
22pF
C2
22pF
Programmer Connect
SCK
MISO
MOSI
RESET
DRAGON_ISP
2x3
3.3V to 5V Buffer
5V_VCC
74ACT125A
74ACT125N
74ACT125B
74ACT125C
C7
100 nF
CLOCK
DATA
UP_TXD
DOTSTAR_LED
5V 1A Regulator
LM2940
1x3
C5
.47uF
C6
22uF
Unused Buffer Port
74ACT125D
Self Leveling Table
Team 04
TITLE: T04_Schematic
Document Number:
REV: 1.3
Date: 12/9/2015 10:01 PM
Sheet: 1/1



12/9/2015 10:50 PM f=4.00 C:\Users\Taylor\Documents\GitHub\ECE-411-Practicum\Eagle\T04_Schematic.brd

Code

- 1400 Lines of C
- More than 50% reused from manuals or examples online.
- Most time consuming.

Code – TWI Communication

- TWI → consists of two signal lines SDA and SCL
- Manual sheet is the main resource
- TWI to enable two way connection between microcontroller and accelerometer
  - Initialize TWI(I2C) in Atmega328 and setup a serial clock to 400kHz
  - Generate TWI start signal
  - Send a byte through TWI
  - Read a byte from TWI, send an acknowledgment back
  - Check status register of the TWI

Code – Accelerometer

- Format of TWI communication (read and write) provided by MMA8452Q Accelerometer user manual
  - Write (initialization): start condition, 0x3A (slave address of device, mode), 0x2A (register to write to – system control register 1), 0x07 (800 Hz sample rate, reduce noise, fast read mode, active), stop condition
  - Read: Start condition, 0x3A (slave address of device), 0x01 (register to start reading from – OUT_X_MSB), repeated start condition, 0x3B (slave address of device, mode), acknowledge each byte, no acknowledge to stop reading

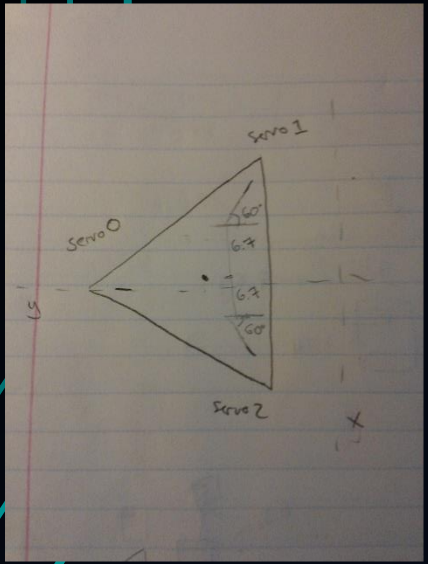
Code – USART

- Used to communicate to Servo controller
- Transmission and initialization functions come from example C code from ATmega328 manual and specifications from Pololu Servo Controller user manual
  - Initialization: set baud rate (9600), set format of messages (8 bit, 1 stop bit, no parity, asynchronous mode), enable transmitter
  - Transmission: make sure transmit buffer is empty, put byte into transmit buffer

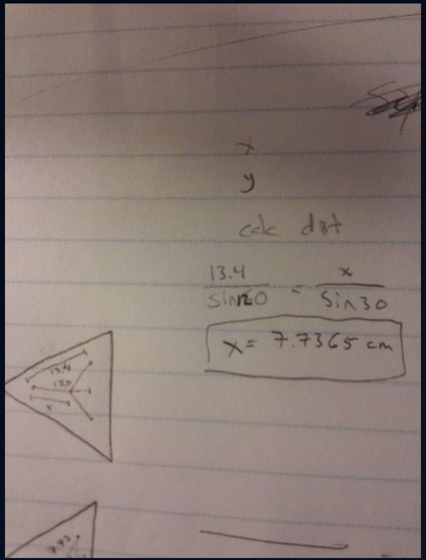
Code – Servo Controller

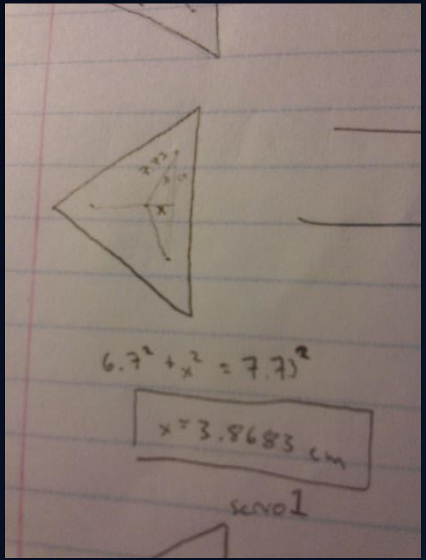
- What needs to be transmit provided by Pololu Servo Controller user manual (SetTarget, SetSpeed, SetAcceleration,SetMultipleTargets)
  - Pololu Protocol: 0xAA (baud rate indication byte), 0x0C (device number), command byte (with most significant bit cleared), command data
  - SetTarget: 0x84 (command), channel number, target low bits, target high bits
  - SetSpeed: 0x87 (command), channel number, speed low bits, speed high bits
  - SetAcceleration: 0x89 (command), channel number, acceleration low bits, acceleration high bits
  - SetMultipleTargets: 0x9F (command), num targets, first channel num, first target low bits, first target high bits, second target low bits, …

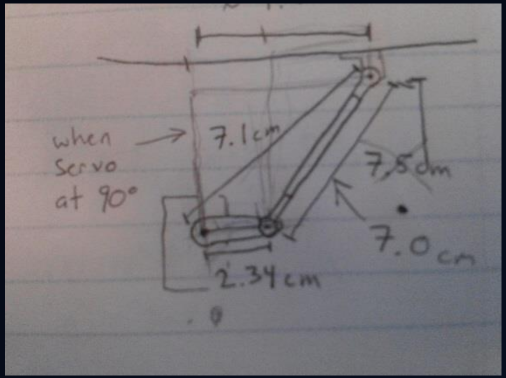
Servo Position Calculation

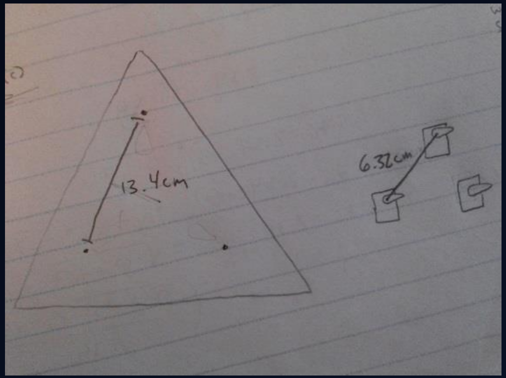


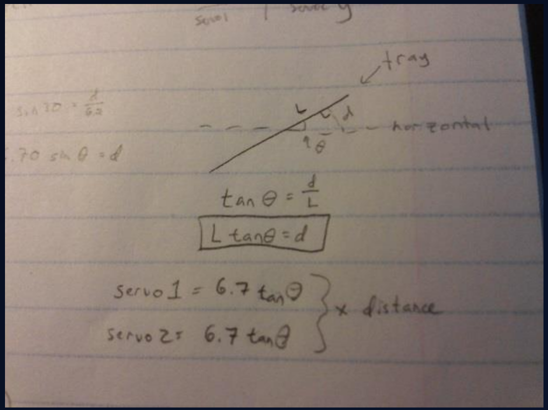
- Consists of 2 parts:
  - Servo angle to tray height
  - Distance the tray is off from the home position

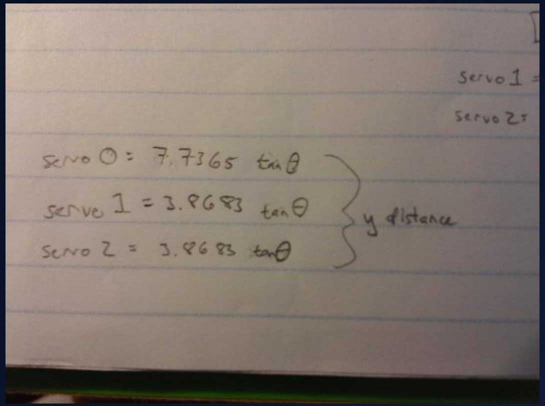






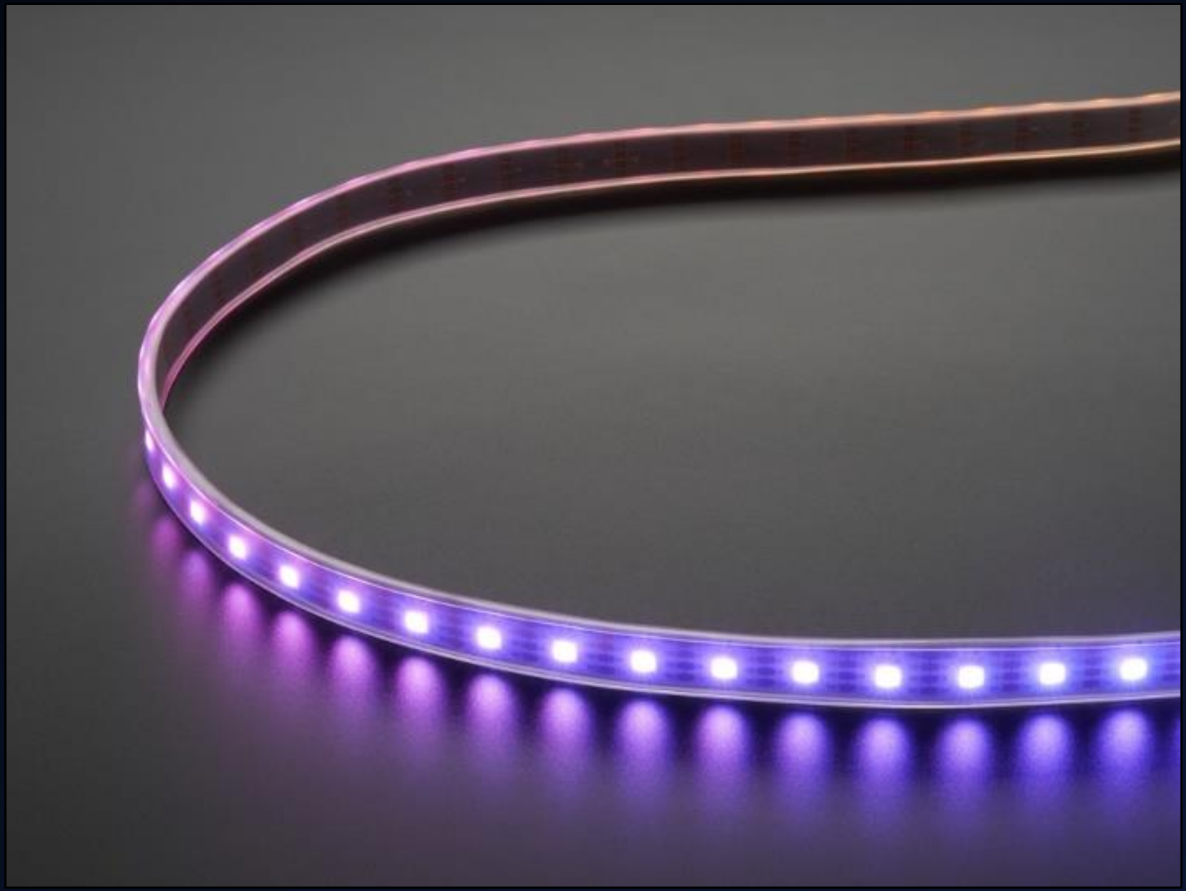






Adafruit Dotstar

- Dotstar vs. NeoPixel
- Easy interfacing
- Adjustable length
- Arduino Library



SPI Communication

- 1 Byte per pixel
- 15 pixels total

Main Program Flow

- Initialize the LED strip
  - Set initial colors
  - output to LED strip
- Initialize the Servo Controller
  - Initialize USART in microcontroller
  - Set speed of servos
  - Set all servos to home position
- Fill Lookup table with servo angle to height mappings
- Initialize the accelerometer
  - Initialize microcontroller for TWI communication
  - Change settings in Accelerometer

Main Program Flow

- Sample accelerometer
  - Sample 400 times and take average
  - Blink the accelerometer communication yellow status LED
- Update the green and red status LEDs
- Servo adjustment
  - Calculate distance off from home position
  - Search through table to find correct servo angle position
  - Update LED strip based on servo angle positions
  - Move servos if they are 4 degrees or more off from previous position
  - Servo movement delay
- Repeat from sample accelerometer

Tools Used

- SolidWorks
- Eagle
- Atmel Studio
  - AVR Dragon Programmer
- GitHub
  - Repository
  - Wiki
- LID
  - Laser Cutting Acrylic
  - Soldering Stations and Oven

IP and Prior Work

- Atmel Code (atmega328 datasheet)
  - TWI Interfacing
  - USART Interfacing
- Internet Search Code
  - BlinkLED (Testing initial setup)
    - electroSome (https://electrosome.com/blinking-led-atmega32-avr-microcontroller/)
  - Binary Search Algorithm
    - Programming Simplified (http://www.programmingsimplified.com/c/source-code/c-program-binary-search)
  - LED Strip Control
    - Adafruit (https://learn.adafruit.com/adafruit-dotstar-leds/overview)
  - Accelerometer Data Conversion
    - Sparkfun (https://github.com/sparkfun/MMA8452_Accelerometer/blob/QC-Rev/Firmware/MMA8452Q_BasicExample/MMA8452Q_BasicExample)
- Arduino
  - Angle – PWM Mapping
    - Arduino Servo Library

IP and Prior Work

- Lots of YouTube videos with examples.
  - https://www.youtube.com/watch?v=ZrZodwEHjew
  - https://www.youtube.com/watch?v=T0SFAdPUUYs
  - https://www.youtube.com/watch?v=5uR34U1qc-Q
  - https://www.youtube.com/watch?v=K-F_T59ZDPw
  - https://www.youtube.com/watch?v=uERF6D37E_o
  - https://www.youtube.com/watch?v=enlQMUE9df4
  - https://www.youtube.com/watch?v=j4OmVLc_oDw

Testing

Functional Test

Materials needed:

Arduino

Arduino IDE

Arduino accelerometer example code

- Verify accelerometer outputs valid measurements
- Verify servos are operable
- Verify Servo controller operates servos
- Verify LED strip operation
- Verify microcontroller operation
- Run full system leveling test

Testing

Integration Test

Materials needed:

Atmel Studio

AVR Dragon Programmer

- Microcontroller to Servo controller (USART) communication
- Accelerometer to Microcontroller (TWI) communication
- Microcontroller level shifting (buffer)

Testing

Parametric Test

Materials needed:

Oscilloscope

Multi-meter

- Output voltage from voltage regulators
- External oscillator frequency
- Range of servos

Testing

Stress Test

Materials needed:

Randomly distributed weights

- Test maximum tray weight

Results

- Most things worked well.
- Servo arms could work better.
- LEDs are nice and bright.

- Rx on Microprocessor from servo controller Tx not stepped down to 3.3V.
- No resets.
- Need better errors and cleaner code.

Contributions

- Sean
  - Mechanical Design, LED Strip control code, GitHub guru.
- Waleed
  - TWI Code, Servo arm assembly.
- Adrian
  - Servo arm design, torque calculations, USART code, servo position algorithm, code formatting.
- Taylor
  - Hardware design, schematic, layout, soldering, LED strip angle color code.
- Everyone!
  - Homework, testing, assembly, coding something.

Lessons Learned

WHAT DID WE LEARN?