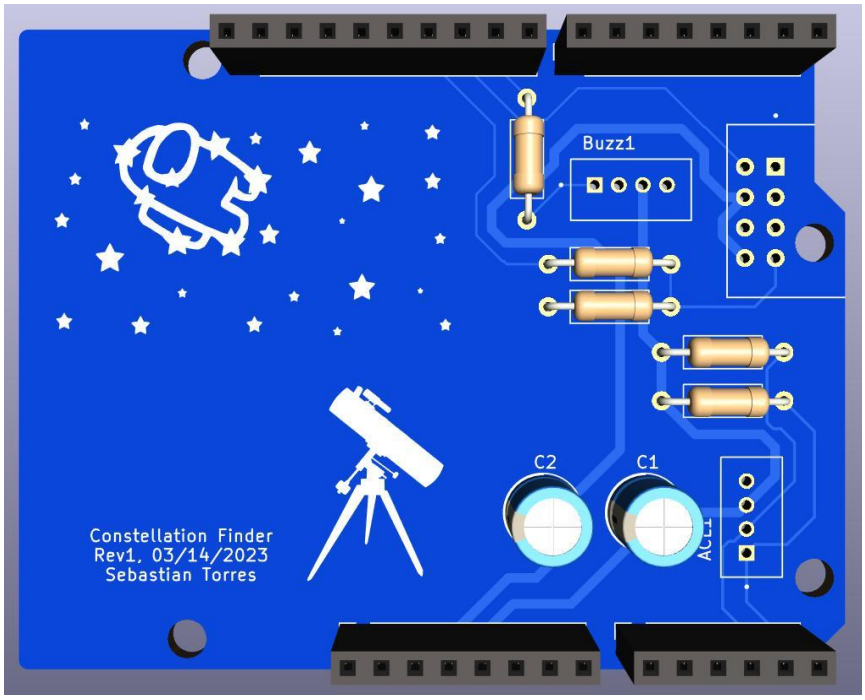
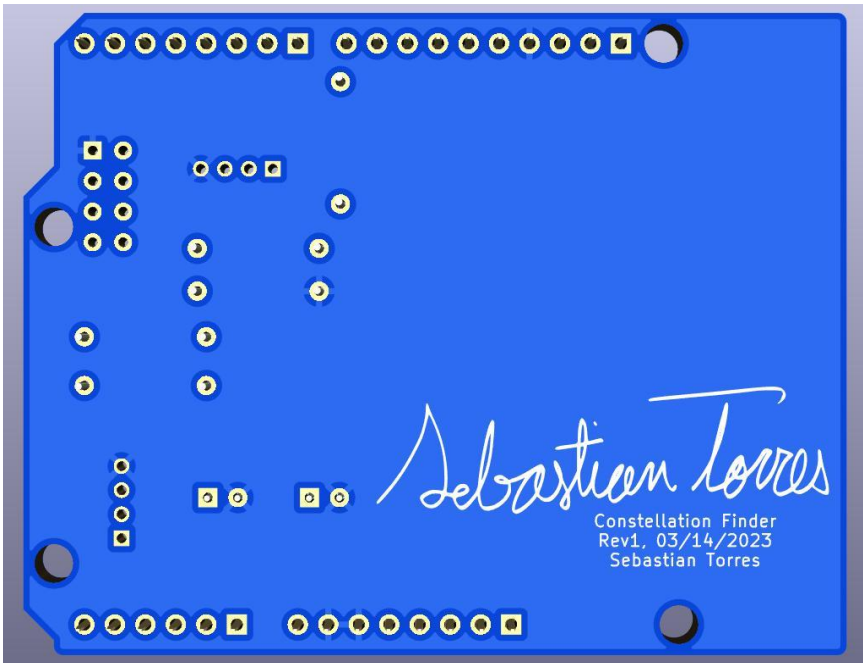
Constellation Finder IoT Device

Sebastian Torres

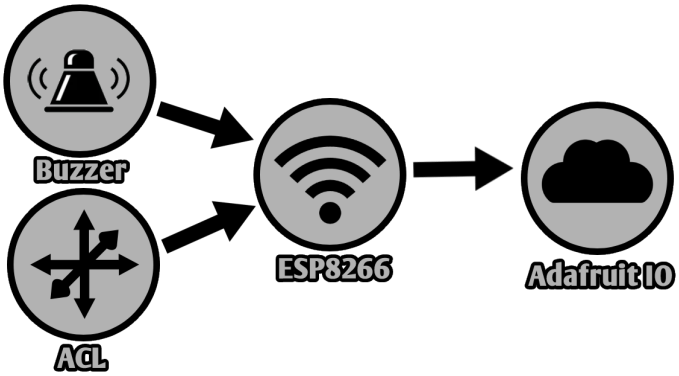




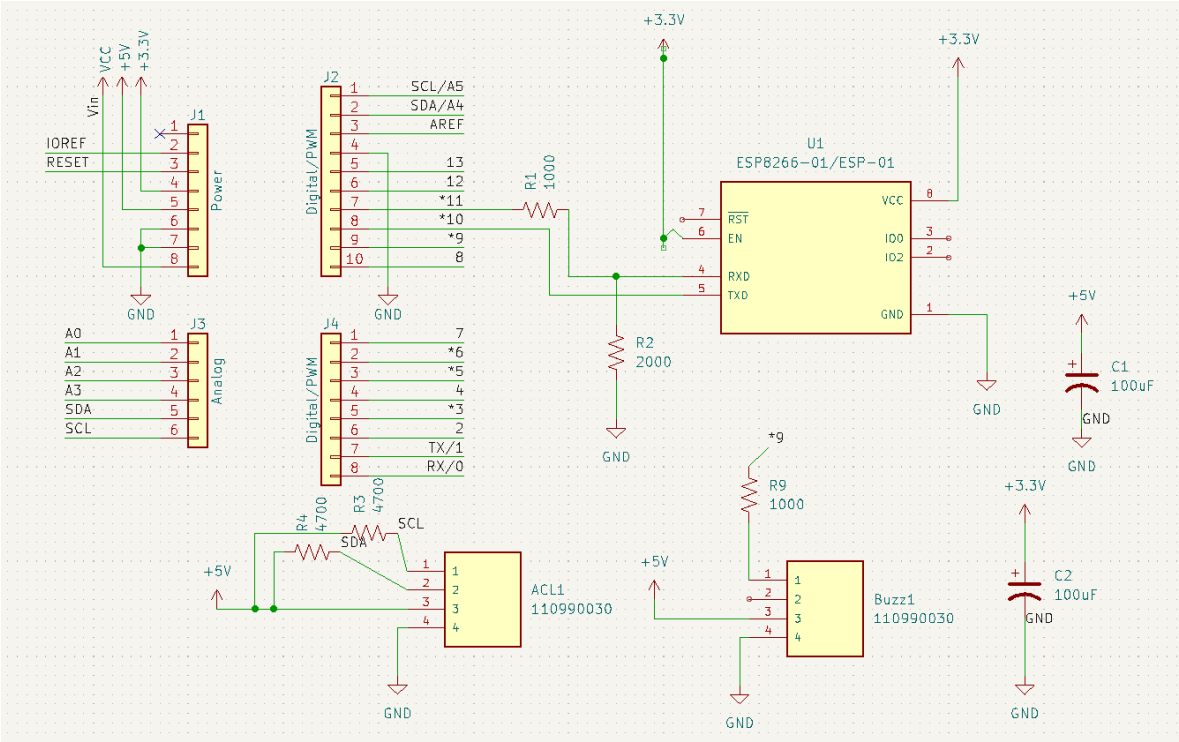
Description

The basic concept behind the circuit is for a connection to be made to Adafruit through the ESP8266, in order to send Accelerometer tilt info to the cloud in order to retrieve constellation data.

A buzzer was also wired to indicate if the device was tilted under the horizon, using a threshold value of 0 on the z-axis of the tilt.

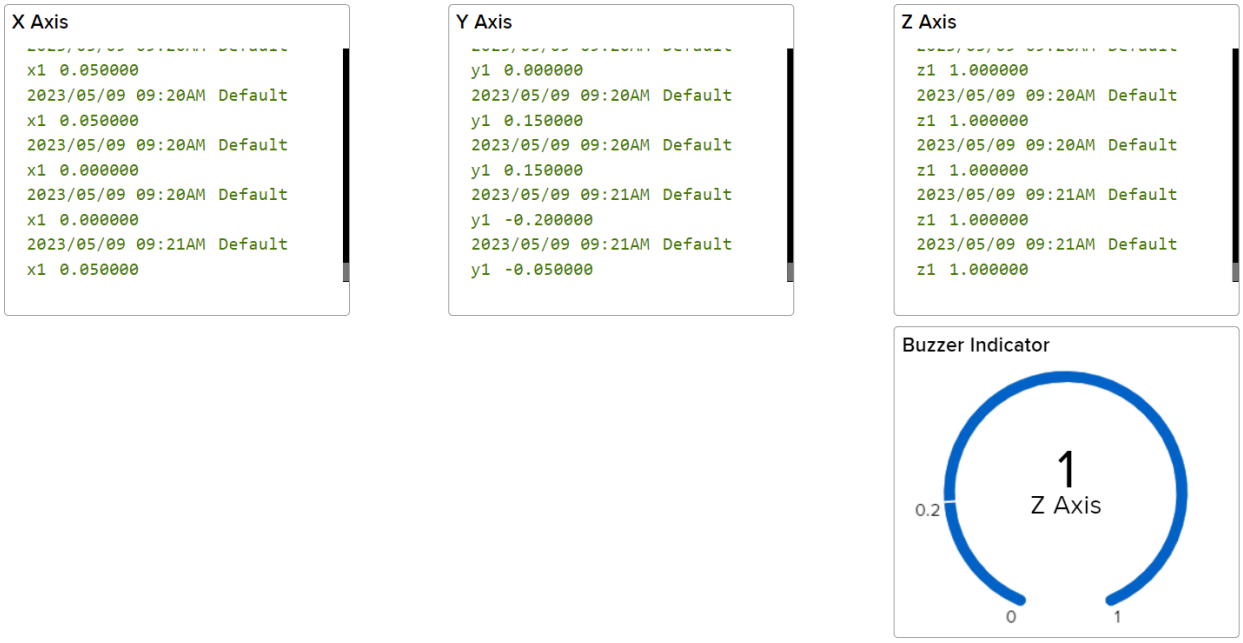


The accelerometer uses an I2C connection in order to communicate with the device, while the buzzer uses a Digital PWM signal pin in order to be turned on, since it is an active buzzer.

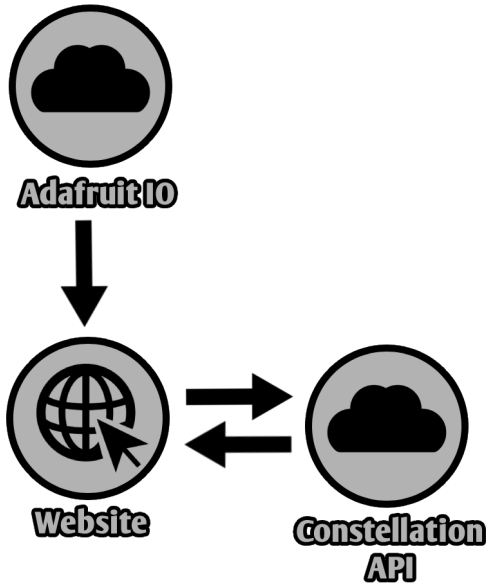


Full Arduino Shield Circuit Schematic

After the accelerometer data is sent to Adafruit, all feed data as well as a visual knob are displayed on a dashboard together as seen below. The knob has a threshold value of .2, with any values read below that changing the color of the wheel to red in order to indicate that the user shouldn't tilt further down.



After the data is sent to Adafruit, the X, Y, and Z axis data is requested from my localhost website using a GET request to the Adafruit API. Once this tilt data is received, it is then processed into celestial coordinates using approximations. After the coordinates are processed, a POST request is made to *Astronomy API* with the coordinates in order to return an image of the stars at the desired location.



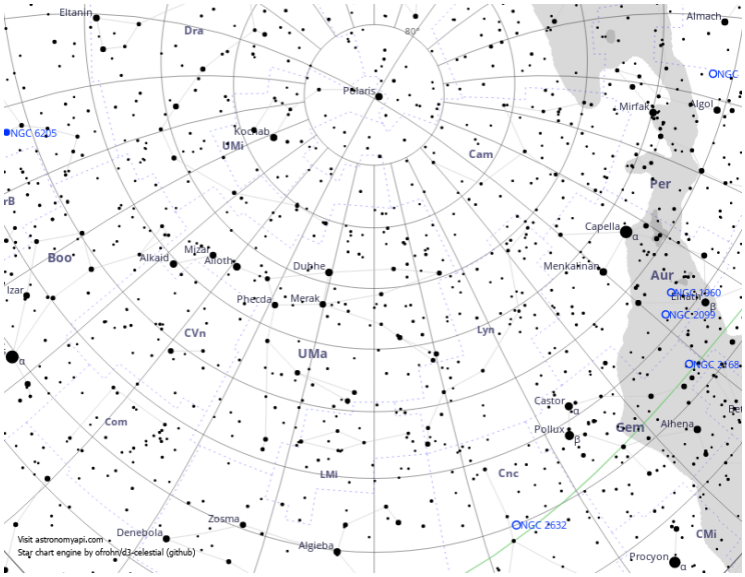
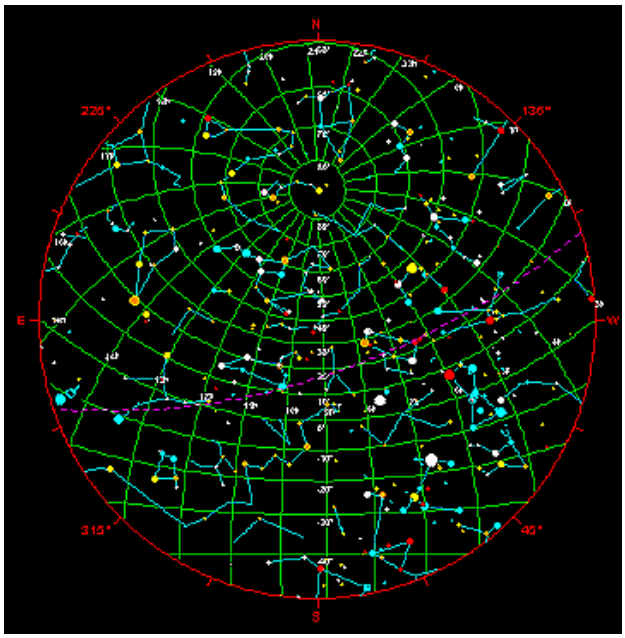


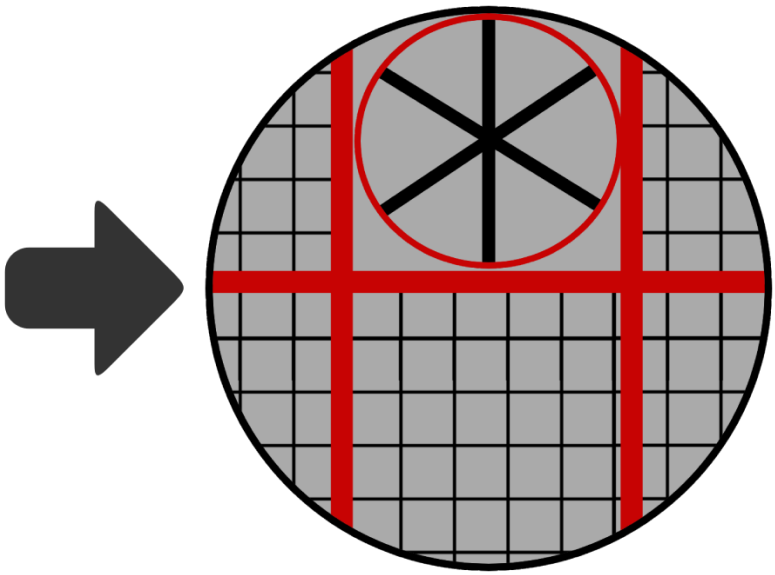
Image returned from Astronomy API

How coordinates are mapped

The main issue encountered with getting an accurate representation of the sky based on tilt was converting values ranging -1 to 1 into celestial coordinates, which are far from linear.



Equatorial star view from the ground



Visualization of approximation used

As seen from the map above, the coordinate system is based on radial lines and circles coming from the north star, Polaris. In order to map the linear coordinates from the accelerometer, I approximated different areas of the visible night sky. The lower half and sides of the circle have grid-like attributes, so I was able to map each of those sections linearly. However, the upper-middle part was much more spherical, so for that part I chose to fix the coordinates to select radial lines coming from the north star. In celestial coordinates, the radial lines are referred to as Right Ascension, while the circles of greater radius coming from the north star are Declination.

I was also able to calibrate the code to depict the changes in stars based on the hour of the day. This is because of the close relationship between Right Ascension and time. Right Ascension values range from 0-23, and as time goes on throughout the day, the Right Ascension values visible at a certain point change proportionally to the time passed, very similar to a clock. For example, if you look at the most northern point on the star map at one time it might have a Right Ascension value of 9. If you look again in an hour, the value there will be 10. Using this property, I was able to notice that in Newark, DE the Right Ascension value in the north will always be the time of the day + 3. For example, at 4pm, the northern Right Ascension value will be about 19 (16 in

military time + 3). However, this will not be true in a few months due to small changes over time.

Project Code

The Arduino Code consisted of setting up the connection to Adafruit through the ESP module. After this, the X,Y and Z tilt values were read from the sensor and sent to each respective feed on Adafruit. A check was also made to see if the Z value was below 1. This meant the device had been tilted over the horizon, which triggered the buzzer to turn on.

The Javascript Code consisted of 3 GET requests made to the Adafruit API in order to retrieve the tilt values that had been sent. After this the values are processed into celestial coordinates (Right Ascension and Declination) using the approximations discussed before. With these coordinates, I then make a POST request to *Astronomy API* and am returned an image of the stars at those coordinates. This image is then displayed on my website.

Bill of Materials

Designator	Footprint	Quantity	Supplier and ref	Purchase Link	Cost
Buzz1, ACL1	SEEED_110990030	2	110990030	https://www.digikey.com/en/products/detail/seeed-technology-co-ltd/110990030/5482560	$3.40
R2	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	2000Ω	https://www.digikey.com/en/products/detail/stackpole-electronics-inc/CF18JT2K00/1741658	$.10
C1, C2	CP_Radial_D6.3mm_P2.50mm	2	100uF	https://www.digikey.com/en/products/detail/nichicon/UVZ1C101MDD/589024?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campaign=PMax%20Shopping_Product_Zombie%20SKUs&utm_term=&utm_content=General&gclid=CjwKCAjw6vyiBhB_EiwAQJRopvymfQHJ-YLC7PhAU-NwkKo7qravxW0Z60Yft4x2tNdnEl-jOGa3UhoCZkIQAvD_BwE	$.58
U1	XCVR_ESP8266-01_ESP-01	1	ESP8266-01/ESP-01	https://www.digikey.com/en/products/detail/universal-solder-electronics-ltd/Ai-Thinker-ESP-01S-ESP8266/14319890	$4.94
R1 ,R9	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	1000Ω	https://www.digikey.com/en/products/detail/stackpole-electronics-inc/CF14JT1K00/1741314	$.20
R3, R4	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	2	4700Ω	https://www.digikey.com/en/products/detail/stackpole-electronics-inc/CF18JT4K70/1741708	$.20

Improvements/Changes

If changing my device, I would add traces on the back of the PCB in order to have a more efficient layout of my through-hole components.

Changes I would make to the code would be to approximate the upper-middle area with spherical coordinates instead of locking it to lines for more accuracy.