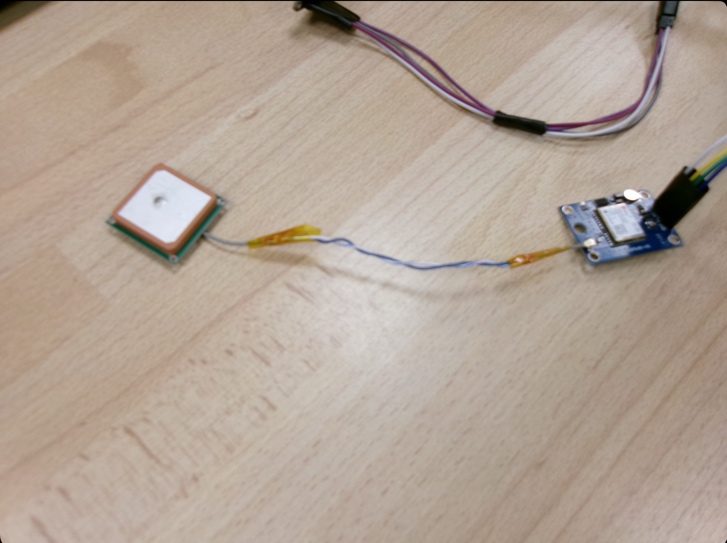
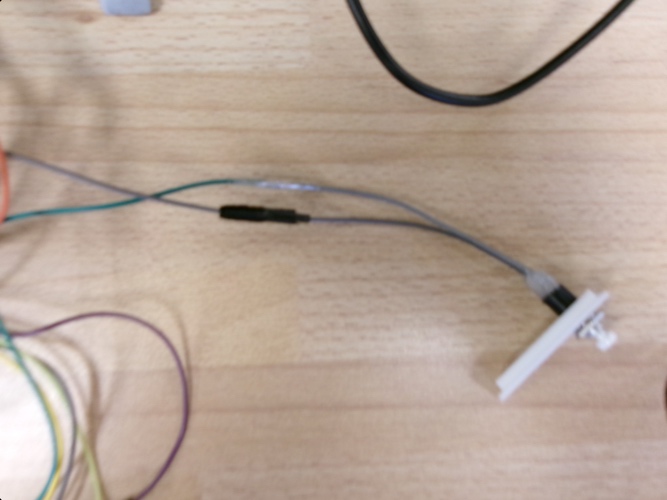
Steps to build

1. To begin we first figured out what we will need for our project. From there we bought the suitable hardware we would need. For example, buying a Raspberry Pi 0 so it could fit in the handle of the walking cane.
2. Once everything had arrived, we began by 3D printing a handle with a compartment on the bottom to store the Pi. This allowed us to place the Pi and vibrating disc closest to the user. The handle was attached to the stick by sliding on then secured with grip tape that wrapped around the handle.<https://www.thingiverse.com/thing:6988852>
3. We then started coding the ultrasonic sensor. We looked at code libraries on Git Hub and were able to create custom code accordingly. To attach the sensor to the Pi we plugged a wire into the 5v plug on the pi and the VCC hole on the Pi. A wire between a ground plug on the Pi and the GND hole on the sensor. A wire between GPIO23 on the Pi and the trig hole on the sensor. And a final wire between GPIO24 on the pi and Echo on the sensor. <https://github.com/SANGERA2/Cyber-Cane/blob/main/home/pi/sensor.py>
4. For the GPS we sourced the inspiration of our code from another Git hub user. Before writing any code, we downloaded the code library from git hub using a specialised code. To set it up we first connected a 3v hole on the pi to the positive voltage plug on the GPS We then connected a ground plug on the pi to 2the negative output on the GPS. Next, we connected GPIO 14 on the pi to the RX hole on the GPS. Last of all we plugged the GPIO15 hole on the pi into the TX hole on the GPS. <https://github.com/SANGERA2/Cyber-Cane/blob/main/home/pi/GPS.py>
5. To set the compass up we first sourced an idea for our code from a page on the Ada fruit website. After writing the base code we used the Atan function to convert the compass bearings into easy to understand 360 bearings. For the wiring we first connected one of the 3v plugs on the pi to the VCC hole on the compass. Next, we joined a GND plug on the pi to the GND plug on the compass. We then joined the GPIO3 plug on the pi to the SCL hole on the compass. Finally, we linked the GPIO2 plug on the pi to the SDA hole on the compass.<https://learn.adafruit.com/lsm303-accelerometer-slash-compass-breakout/coding> And to calibrate the compass you can use <https://github.com/SANGERA2/Cyber-Cane/blob/main/home/pi/calibrate.py>
6. For the buzzers that would give us our bearings, distance, and degrees to target on command we attached one of the two wires to GPIO19 and Ground pin 49. We did the same with the second buzzer with GPIO13 and Ground pin 34 respectively. We created our own code for the vibrating discs, however there are websites such as Ada fruit or Forum that give you very simple code.<https://forum.arduino.cc/t/mini-vibration-motor/619895>
7. Next, we attached the button using the plugs on the pi using GPIO12 and Ground 30. We were able to combine inputs and outputs using code we made. This allowed us to get bearings, distance to an obstacle, and in what direction an obstacle is in. A good tutorial explaining how to do this is on RasPI TV. <https://raspi.tv/2013/rpi-gpio-basics-6-using-inputs-and-outputs-together-with-rpi-gpio-pull-ups-and-pull-downs>
8. One of our last additions was the Espeak. This meant that the canes user can listen to all the outputs the cane gives you. We did this by looping a section of code that reads out text. A good website that gives you some simple and easy to follow code for the Espeak is Git Hub. We also were able to utilize the Bluetooth function that Pi’s have built in. This means the user can connect a pair of wireless speakers to the cane.<https://github.com/espeak-ng/espeak-ng>
9. To get our code to be running automatically when the cane is turned on, we had to import *time* and then put all of our code into a loop. [This](https://github.com/SANGERA2/Cyber-Cane/blob/main/lib/systemd/system/sensor.service) Is a guide for the automatic start of the sensor, and [this](https://github.com/SANGERA2/Cyber-Cane/blob/main/home/pi/.config/systemd/user/gps.service) Is our code for navigation to automatically start.
10. To finally set the whole thing up our teacher helped us solder the wires onto the Raspberry Pi 0. We then taped the sensor to the end of our cane and placed the rest of the pi in the handle while letting the GPS hang out on our specifically designed clip so it could take directions. And that was the end, after multiple months of hard work our final design was done, and we were able to film.

