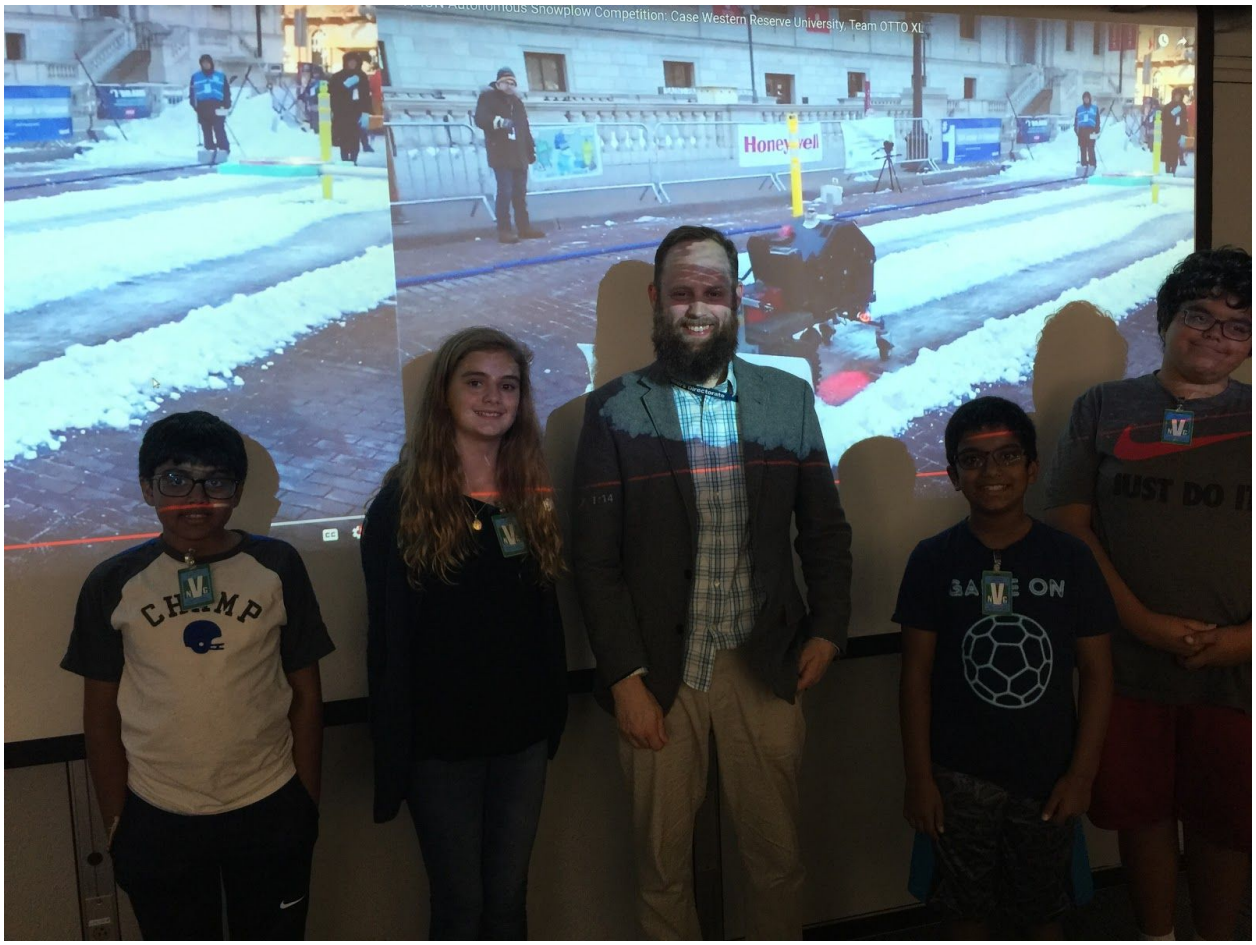
PROJECT

Team STEVE likes robotics so much we tied robotics into our project this year. Our project this year falls around saving the water organisms, we learned about preventing Deoxygenation (which kills fish). We made a prototype robot that automaticity plows snow and spreads environmentally friendly snow products. The reason we need to spread certain salts is because traditional rock salt, when it falls into the water, it removes the oxygen a process which is called deoxygenation. The fish can no longer breath the oxygen from the water and they die. A lot of this rock salt gets into our water because people put an excess amount of salt on their driveways. This is a big problem in the Springboro community especially because Springboro gets most of our water from underground springs (hence the name **Spring**boro) the salt leaking into our water source is even worse. Most new products now either use natural chemicals that melt the snow or salt, enough to melt the snow, but not enough to cause deoxygenation. Many people put rock salt on their driveways or paths, not knowing the consequences.

After discovering the rock salt/deoxygenation problem team STEVE! traveled to WPAFB and visited Dr. Donald Venable in the Sensors Directorate who is a Navigation Engineer.



Dr. Venable taught us about robot navigation. We learned about GPS and talked about a autonomous snow plowing competition for college students that Dr. Venable helps to officiate for the Institute of Navigation:

http://www.autosnowplow.com/welcome.html

We researched about the things we talked about and soon discovered that these things were 1: out of our budget (Which wasn't very big we made our robot out of recycled materials) 2: We didn't have the time/resources to put it together. So we decided that we could still use some of Dr. Venable's lessons to our advantage, so we still did the robot plow idea but not in the exact way we had planned.

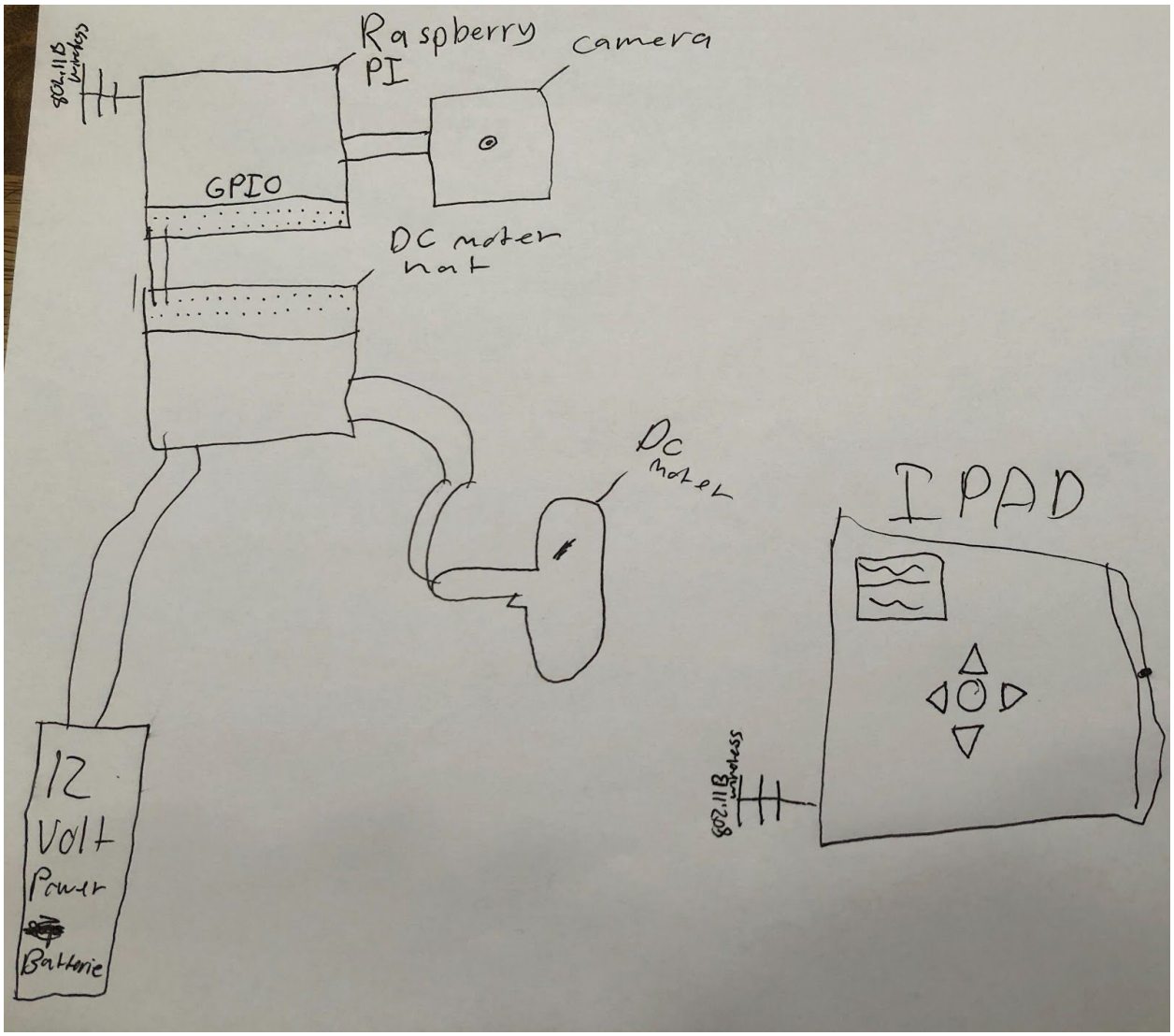
Our project robot is called the **A**utomatic **R**obot **T**hat **P**lows **S**now and **S**hoots **O**ut **E**nvironmentally **F**riendly **M**elting **S**now **S**tuff (ARTPSASOEFMSS). We divided the project into parts so it could be worked on simultaneously by multiple team members. Elaina did the research on the project and discovered the problem. Jordan donated the "power wheel" which was a Thomas the Train he used to play with when he was younger. Dallen and Coach Burks did the mechanical work, painting, wood cutting, and put the snow shovel on the front of the robot. While Elijah and Coach Rovito did the electronics for the ARTPSASOEFMSS. Finally Snath did the project posters and anything else we needed (kinda just someone we could throw stuff to do at):

MECHANICAL

We started with a Power Wheel toy that looks like “Thomas the Train”. The Power Wheel came with a powerful DC motor which can move 50 pounds, wheels, battery charger, and a 12 volt 4 amp hour battery. To make the ARTPSASOEFMSS prototype significant mechanical work had to be completed like removing the seat and finding then installed a spreader mechanism. In addition a platform was built to hold the electronics and spreader. Finally a shovel was added to act as a plow for the robot. Here is a picture of Dallen and Coach Burks hard at work on building the mechanical system for the ARTPSASOEFMSS robot:

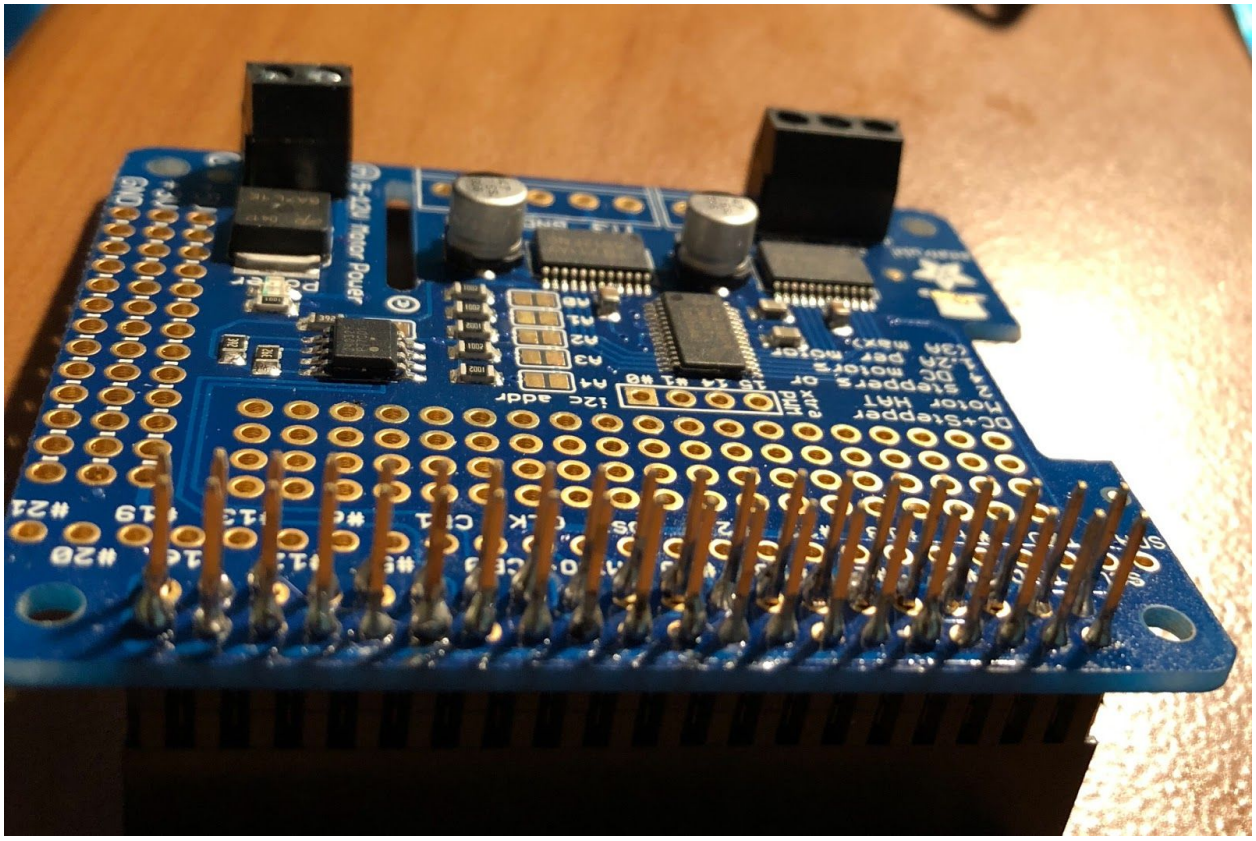
ELECTRONICS

ARTPSASOEFMSS uses a Raspberry Pi which is an embedded computer about the size of a credit card and uses a similar processor to a modern smartphone. The Raspberry Pi runs the Linux operating system and is extremely programmable. The major components and a schematic is provided below:

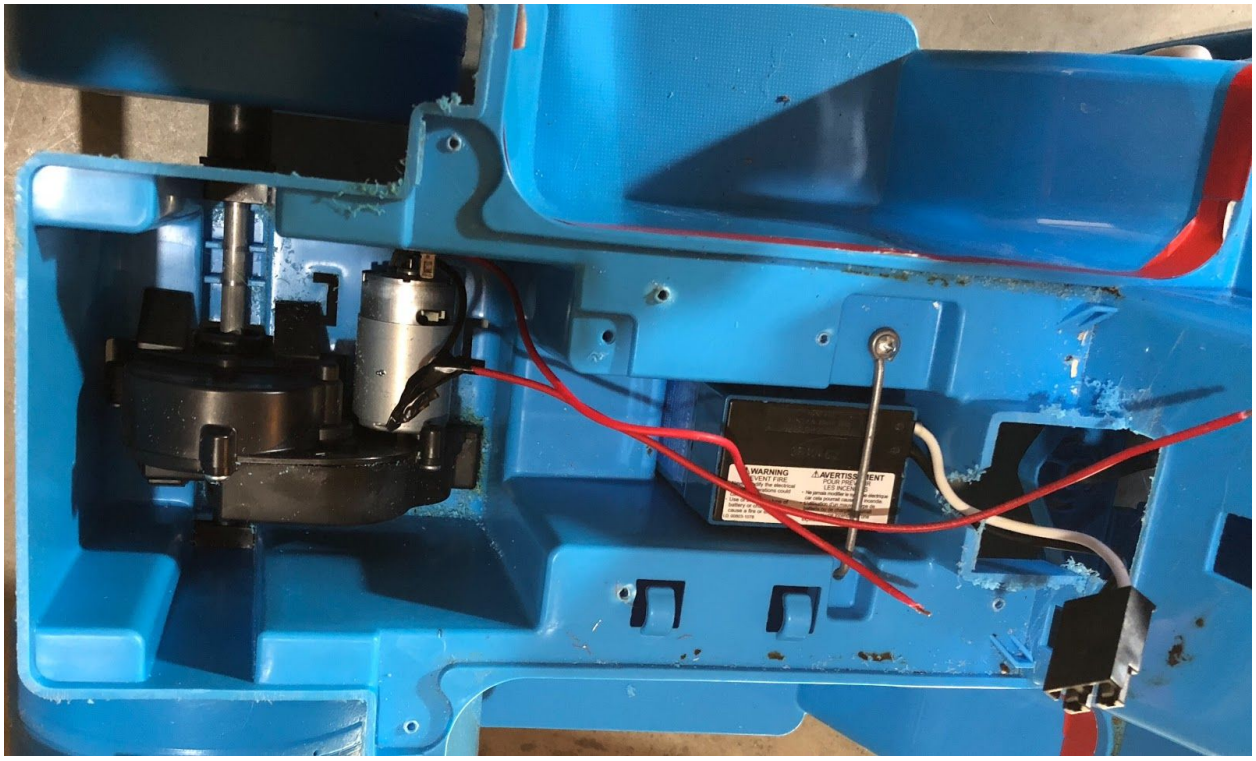


A small company in New York called adafruit.com makes hardware attached on top (HAT) devices to extend the functionality of the Raspberry Pi. These HAT boards have to be soldered so Elijah learned how to solder:

Soldering is joining two metals in such a way that electrons can then flow between the metals and complete a circuit. The soldering irons can get up to 450 degrees so extreme caution has to be applied. Elijah burned himself three times making this robot but it was worth it. Having the right solder iron and the proper solder is critically important for delicate electronic work. The Raspberry Pi has General Purpose IO (GPIO) pins that is used to communicate from the Raspberry Pi to the HATs. The GPIO pins consistent of 40 pins so it requires patience and skill.

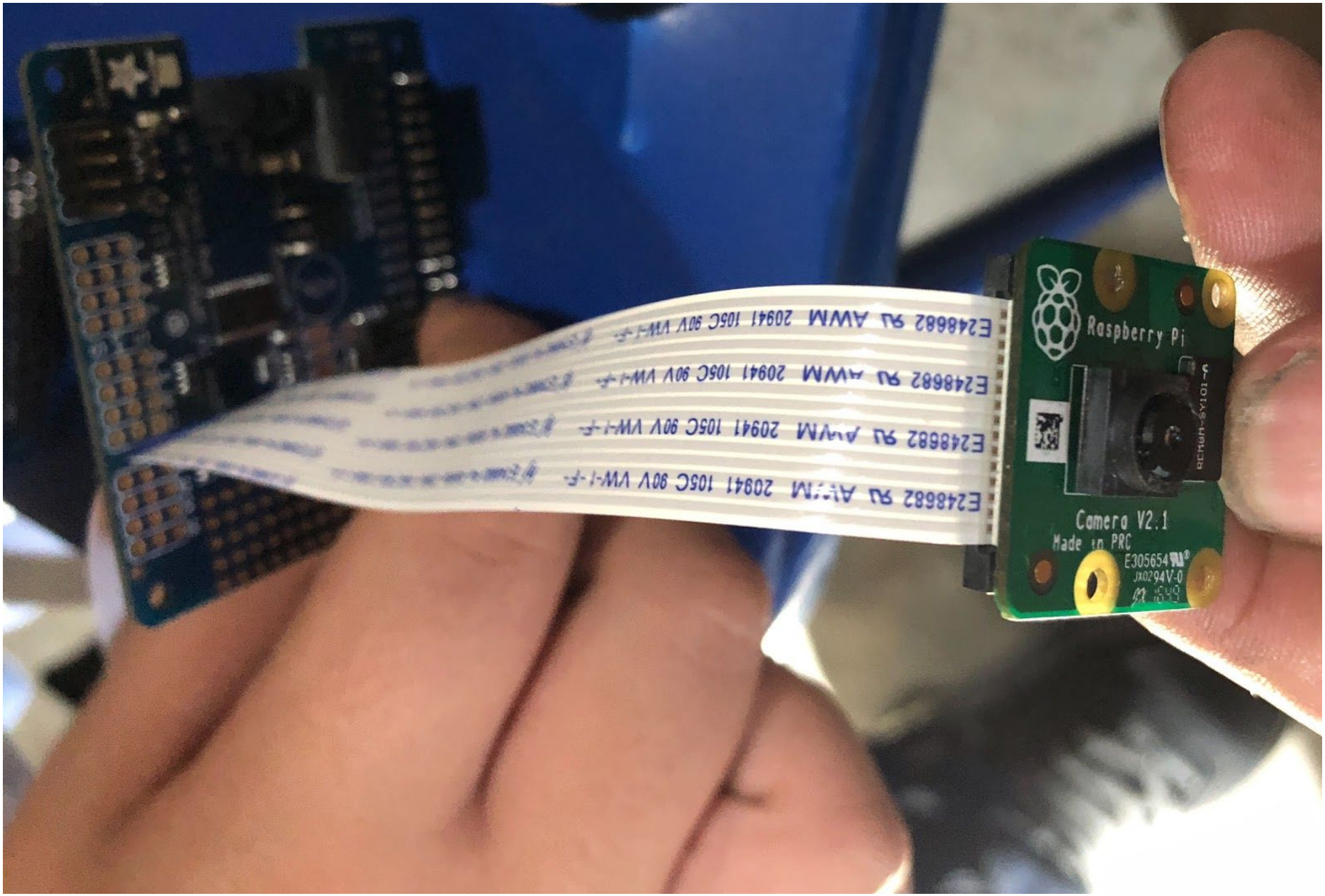


In the picture above you can see the precision required to solder these pins, if too much solder is applied then the signals that flow over the pins will be mixed and the HAT will not function. The HAT pictured above controls a DC motor with a popular technique for control called pulse width modulation (PWM). The length of the pulse tells the motor how fast to go and in which direction forwards or backwards. Elijah had to extend the wiring inside of Thomas the Train to be able to connect the wires to the Raspberry Pi. This was done with a soldering iron and electrical tape. Notice the wire inside of Thomas the Train with the black electrical tape.



Since we did not buy the DC motor we know very little about the motor. Dallen and Coach Burks removed the motor and discovered that it had no writing or markings on the motor which would allow us to look up the motor specifications. The motor THAT has a 3 amp peak limit so we drive the ARTPSASOEFMSS slowly to make sure we don't have a thermal failure. One thing we could do is design a circuit that goes in between the battery and motor HAT to make sure that the system doesn't draw more than 3 AMPS, perhaps a future prototype feature we could add?

In order to control this robot a camera streams to your controlling device and it allows you to control the robot. The camera hooked up to a raspberry PI streams over the web to your device (iPad, iPhone, etc). The user can then control the robot from the user's device in the comfort of their home or wherever they want as long as it's in a reasonable range. Camera's resolution and frame rate can be controlled in case of wireless network limitations.


Raspberry Pi
Camera V2.1
Made in PRC
E305654
94V-0

COST BREAKDOWN FOR
ARTPSASOEFMSS

ITEM/DESCRIPTION	***COST***
Motor Hat	$22.50
Stacking Header (extra tall)	$2.95
Brass Standoffs	$3.00
4 x AA battery holder	$2.95
Servo Hat	$17.50
High Torque servo	$19.50
Flex Cable (PI CAMERA 24”)	$2.95
RASPBERRY PI 3	$35.00
Thomas Tank engine	Donated ($244)
Total	$106.35(without power wheel) $350.35(with power wheel)

PROGRAMMING

Programming for the ARTPSASOEFMSS was done in Python with team STEVE using the examples provided with the hardware. Python is a high level language designed to be easy to use, it has extensive usage in science and engineering. The Raspberry Pi camera has a provided Python library with very good examples which helped us to setup the camera and get it streaming quickly. Below is the code that we used for the streaming camera:

```
'''
This demo code came from:
    http://picamera.readthedocs.io/en/release-1.13/recipes2.html#web-streaming
'''
import io
import picamera
import logging
import socketserver
import datetime as dt
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>picamera MJPEG streaming demo</title>
</head>
<body>
<h1>PiCamera MJPEG Streaming Demo</h1>
<img src="stream.mjpg" width="640" height="480" />
</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
```

```
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=10) as camera:
    output = StreamingOutput()
    camera.vflip = True
    camera.hflip = True
    #camera.annotate_background = picamera.Color('black')
    #camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()
```

FUTURE WORK

Future prototype improvements include:

- making the ARTPSASOEFMSS autonomous which means no human control

- be able to spread liquids so that is doesn't have any salt involved
- being able to make the robot turn
- Better user interface

Some of these things we were planning to add and ran out of time! But if we were to advance we could do some of these things (wink wink).