Task 2
Robot Agriculture Monitoring System

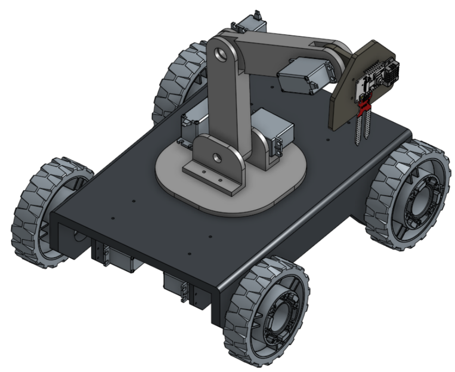
Nathan Yohannes
100749914

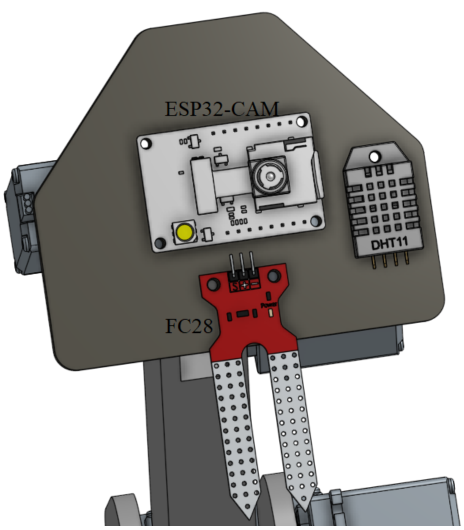
26 May 2022

CAD Model:
https://cad.onshape.com/documents/b062de93db733fabd56846bc/w/1eb1cf6a3f9794be1980c31e/e/513cbd3c1d2c6e78a2664ea1?renderMode=0&uiState=628fbce34dc8c5165faff984

Robot Agriculture Monitoring System

The Robot Agriculture Monitoring System (RAMS) is a self contained IOT Device intended to monitor the status of protected plants in a closed environment. The system was designed to operate in remote or intentionally vacant domains by updating supervisors with information about the plant life and their environment. The RAMS excels in such spaces by autonomously sampling and delivering information specific to each plant to offsite supervisors. The RAMS may also be controlled by a remote operator via the accompanying web application, allowing for better surveillance and information gathering.





The RAMS is able to send and receive information over Wi-Fi via the equipped ESP32-CAM microcontroller. This enables the RAMS to send the data gathered on each testing cycle, or be controlled remotely should further investigation be necessary. Attached to the RAMS' arm is the camera module of the ESP32-CAM, the DHT11 humidity sensor, the FC28 soil moisture sensor, and an optional TFmini Plus LiDAR Range Finder for environment mapping. Excluding LiDAR, the sensors work independently to gather their respective data about each plant before the microcontroller packages the information to be reviewed.

The RAMS uses two different methods in order to path between the goals (plant basins). These are either using LiDAR to model the environment and using the RRT* to path plan, or having a robot operator create a path for the robot to use. With a path generated, RAMS positions itself within a distance from the plant basin using the QR Code as a reference. Once in position, the robot arm deploys and reads samples from the plant and moves on to the next goal.

```
desired_w;
desired_h;
margin;
procedure move_into_position(w,h):
    error_w = desired_w - w;
    error_h = desired_h - h;
    if(error_w > margin AND error_h > margin):
        t = get_heading_to_goal();
        turn(t);
        while(error_w > margin OR error_h > margin):
            drive(low_speed);
        drive(stop);
```

Pseudo Codes

```
desired_w;
desired_h;
margin;
procedure move_into_position(w,h):
    error_w = desired_w - w;
    error_h = desired_h - h;
    if(error_w > margin AND error_h > margin):
        t = get_heading_to_goal();
        turn(t);
        while(error_w > margin OR error_h > margin):
            drive(low_speed);
        drive(stop);
```

```
procedure send data()  returns boolean:
    SMTP_Message message;
    SMTP_Attachment attachment;
    attachment.file.path(path_to_attachment);
    message.addAttachment(attachment);
    if(!smtp.connect(&session)):
        print(“couldn’t connect to mail server”);
        return False;
    if(!MailClient.sendMail(&smtp, &message)):
        print(“couldn’t send email);
        return False;
    return True;
```

```
procedure connect_to_wifi(ssid, password) returns boolean:
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while(WiFi.status !=  connected):
        print(‘.’);
        delay(1000);
```