Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Team 41: Roombotics
Bi-Weekly Update 3

James Deere, Taylor Mosser, Todd Van Klaveren
Sponsor: Eric Robles
TA: Max Lesser

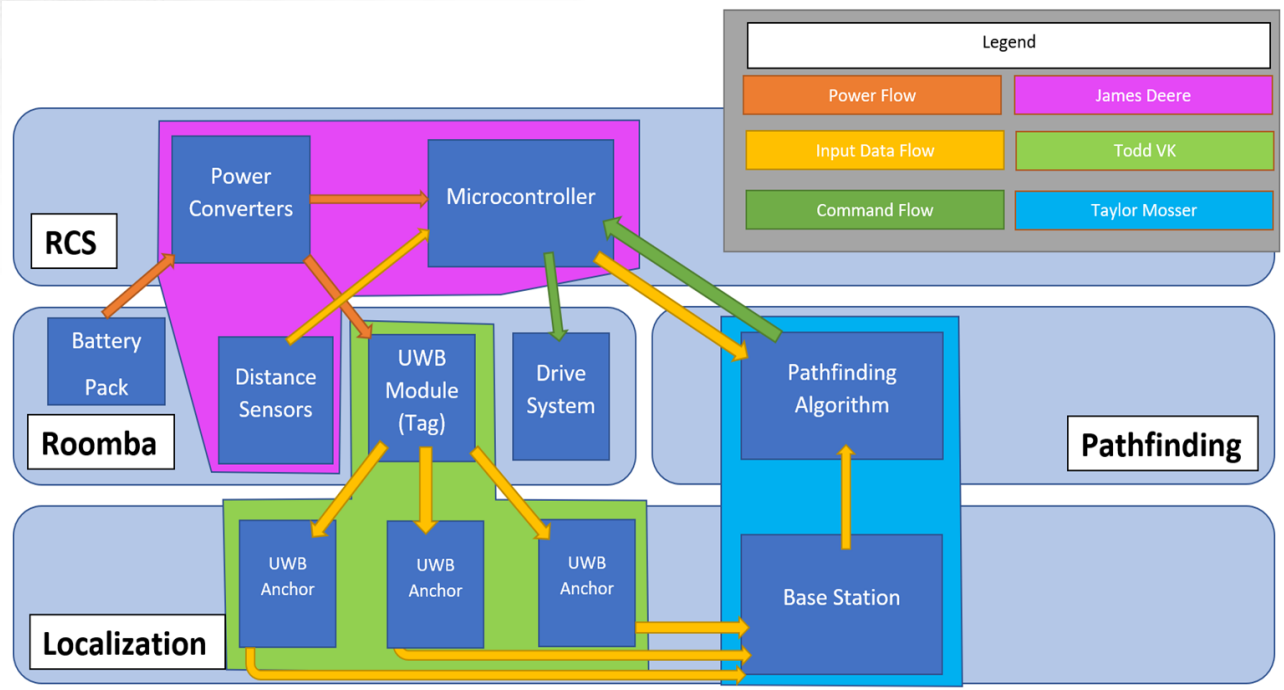
Roomba-MRS Project Summary

Problem: iRobot Roombas do not follow a specific path when cleaning a room and are prone to repeatedly cleaning the same areas or avoiding other areas altogether. While the Roombas can map the layout of a room, they do not have strategic methods of navigating the space to clean quickly and efficiently.

Additionally, one Roomba can be ineffective in cleaning larger spaces due to its small size. This system will allow for more than one Roomba to be deployed in a space in which they can work together to cover all area in the room.

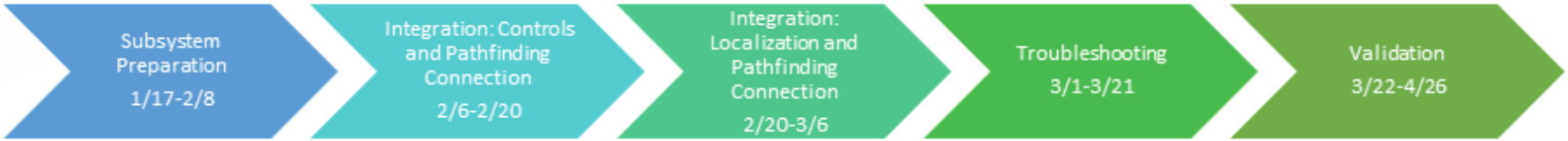
Solution: Roomba will navigate a space using the A* pathfinding algorithm to calculate lowest cost path from one point to another. Path calculation will be performed based on the location of the Roomba and any obstacles in the room. Roombas will not interfere with each other's assigned cleaning spaces and will avoid colliding with each other.

Project Overview



- Pathfinding Algorithm decides best path for each Roomba
- Base Station sends movement commands via Bluetooth
- Localization determines actual location of Roombas
- Feedback from localization updates Pathfinding algorithm.

Project Timeline



Pathfinding and controls subsystems have been integrated. Currently integrating pathfinding and localization.

Roomba Control Subsystem

James Deere

Accomplishments since last update 15 hrs of effort	Ongoing progress/problems and plans until the next presentation
• PCB is fully assembled • PCB tested and works as designed • Integrated controls with pathfinding subsystem via bluetooth connection	• Smoothing out turning in control firmware for better accuracy • Reintroducing obstacle detection with 5 distance sensors to integrated control/pathfinding subsystems • Ongoing learning of 3D CAD software to elegantly mount PCB, sensors, and localization module onto Roomba • Attend Blitz

Roomba Control Subsystem

James Deere

Pathfinding

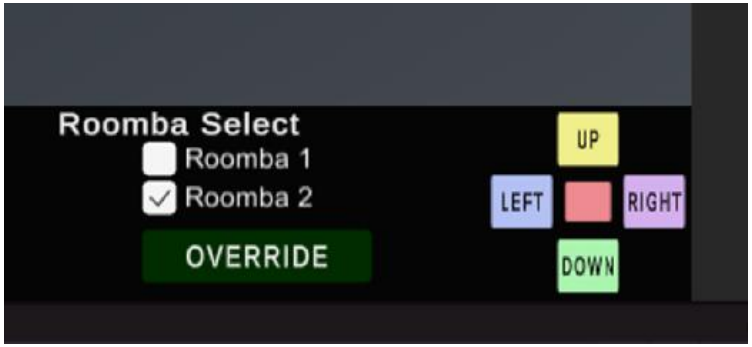
Taylor Mosser

Accomplishments since last update 20 hours of effort	Ongoing progress/problems and plans until the next presentation
- Pathfinding and controls integrated; can send commands through Unity to ESP32 with bluetooth connection - Unity UI completed - Manual control of pathfinding for simulation finished	- Integrate pathfinding and localization subsystems - Create scene/game view for real-life system - Test commands in Unity with Roomba

Pathfinding

Taylor Mosser

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class UpButton : MonoBehaviour
{
    public Button m_UpButton, m_DownButton, m_LeftButton, m_RightButton, m_StopButton, m_OverrideButton;
    public GameObject rmba;
    public GameObject toggler;


    private float speed = 5.0f;
    private bool goUp = false;
    private bool goDown = false;
    private bool goLeft = false;
    private bool goRight = false;
    private bool goStop = false;
    private bool overrideStatus = false;


    void Start()
    {

        m_UpButton.onClick.AddListener(TaskUp);
        m_DownButton.onClick.AddListener(TaskDown);
        m_LeftButton.onClick.AddListener(TaskLeft);
        m_RightButton.onClick.AddListener(TaskRight);
        m_StopButton.onClick.AddListener(TaskStop);
        m_OverrideButton.onClick.AddListener(TaskOverride);
    }

    void TaskUp()
    {
        Debug.Log("You are definitely, certainly moving up!");
        goUp = true;
        goDown = false;
        goLeft = false;
        goRight = false;
        goStop = false;
        Update();
    }

    void TaskDown()
    {
        Debug.Log("You are moving down!");
        goDown = true;
        goUp = false;
        goLeft = false;
        goRight = false;
        goStop = false;
    }

    void TaskLeft()
    {
        Debug.Log("You are moving left!");
        goLeft = true;
        goUp = false;
        goDown = false;
        goRight = false;
        goStop = false;
    }

    void TaskRight()
    {
        Debug.Log("You are moving right!");
        goRight = true;
        goUp = false;
        goDown = false;
        goLeft = false;
        goStop = false;
    }

    void TaskStop()
    {
        Debug.Log("You have stopped!");
        goStop = true;
        goUp = false;
        goDown = false;
        goLeft = false;
        goRight = false;
    }

    void TaskOverride()
    {
        Debug.Log("Override clicked!");
        overrideStatus = true;
    }



    // Update is called once per frame
    void Update()
    {
        if ((goUp) && (!goDown) && (!goRight) && (!goLeft) && (!goStop) && (toggleChoice(toggler.GetComponent<Toggle>().isOn)) && (overrideStatus))
        {
            transform.Translate(0.05f * Time.deltaTime * speed, 0f, 0f);
        }

        else if((goDown) && (!goUp) && (!goRight) && (!goLeft) && (!goStop) && (toggleChoice(toggler.GetComponent<Toggle>().isOn)) && (overrideStatus))
        {
            transform.Translate(-0.05f * Time.deltaTime * speed, 0f, 0f);
        }

        else if((!goDown) && (!goUp) && (!goRight) && (goLeft) && (!goStop) && (toggleChoice(toggler.GetComponent<Toggle>().isOn)) && (overrideStatus))
        {
            transform.Translate(0f, -0.5f * Time.deltaTime * speed, 0f);
        }

        else if((!goDown) && (!goUp) && (goRight) && (!goLeft) && (!goStop) && (toggleChoice(toggler.GetComponent<Toggle>().isOn)) && (overrideStatus))
        {
            transform.Translate(0f, 0.5f * Time.deltaTime * speed, 0f);
        }

        else if ((!goDown) && (!goUp) && (!goRight) && (!goLeft) && (goStop) && (toggleChoice(toggler.GetComponent<Toggle>().isOn)) && (overrideStatus))
        {
            transform.Translate(0f * Time.deltaTime * speed, 0f * Time.deltaTime * speed, 0);
        }
```



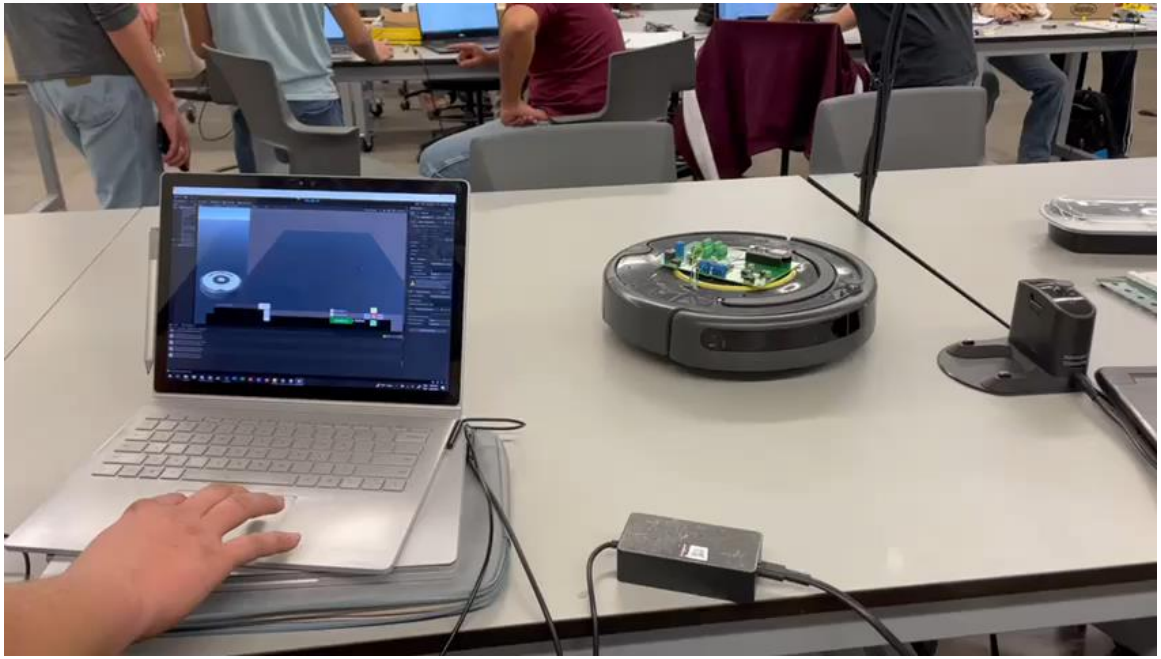
Localization/Integration Design

Todd Van Klaveren

Accomplishments since last update 20 hrs	Ongoing progress/problems and plans until the next presentation
Integrated Bluetooth library into Unity code. Unity application can now: • Scan for Devices • Initiate pairing • Connect to Roomba Controls Device by name or ID • Issue Commands to Roomba Control Unit • Listen for any responses from the Roomba	Testing each command and the response rate of the Roomba. (Tonight in Lab) Expose methods of the Bluetooth control subset to the application at large. Integrate pathfinding AI by teaching it the methods we expose for sending and receiving messages. Integrate serial data stream from localization initiator anchor.

Localization Subsystem

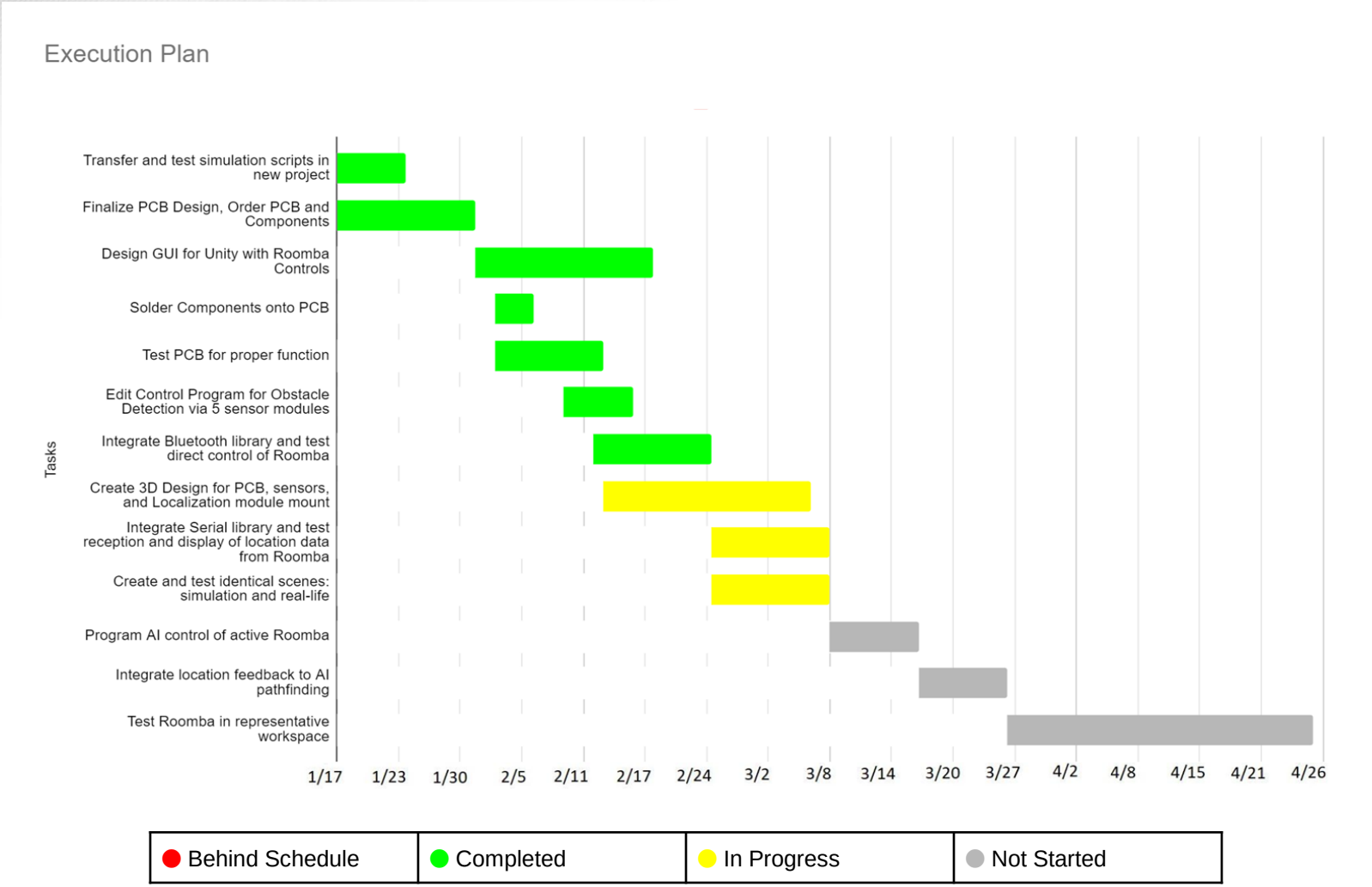
Todd Van Klaveren



Parts Ordering Status

- All parts have arrived, no more are needed at this time

Execution Plan



Validation Plan

FSR	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
3.2.1.6	Object Detection	Roomba can detect an object in front of it (PCB and 5 sensor configuration).	Obstacles will be placed in front of Roomba at various sizes and angles to determine if there are blind spots.	UNTESTED	James Deere
3.2.3.3	Voltage Converters	The two converters used on the PCB output 3.3V at 1 amp and 5V at 750 mA respectively	A multimeter and E-load is used to check the output voltage and current levels.	SUCCESS	James Deere
4.1.3	Mounting	The PCB, UWB module, and sensor modules shall be mounted onto the Roomba in a sleek and elegant way, allowing for no blind spots.	Mount adds less than 50% to the height of Roomba and allows for no blind spots.	UNTESTED	James Deere, Todd Van Klaveren
3.2.1.5	Command Response	Roomba can receive commands from the base station and respond accordingly.	A set of known commands will be flashed to the MCU from the base station via Bluetooth	SUCCESS	Team
3.2.1.6	Object Response	Roomba stops and sends an alert of the obstacle's presence to the base station.	Obstacle will be placed in Roomba's commanded path; Roomba should stop once obstacle is detected and send Bluetooth signal alerting the base station that there is an obstacle.	UNTESTED	Team
3.2.1.7	Localization	Localization system reports Roomba location to Base Station	Base Station Application will output location data. Data will be compared to measurements of Roomba. Roomba must be within 13" circle of reported.	UNTESTED	Taylor Mosser, Todd Van Klaveren
3.2.1.5	Roomba Control	Base station manual control can issue commands to Roombas individually.	Roomba must execute commands being sent from base station application inputs made using the GUI.	UNTESTED	Team
3.2.1.1	AI Control	Pathfinding AI can issue commands to Roombas individually.	A* algorithm empty object and AI pathfinding script can be used in multiple scenes without interference. Commands to real-life Roombas can be given using Unity GUI.	UNTESTED	Taylor Mosser
3.2.1.1, 3.2.1.2, 3.2.1.6, 3.2.1.7, 3.2.4.4	In use Testing	Roomba navigates the test environment cleanly (no bumps) and updates path according to discovered obstacles.	A successful test will have no collisions between Roombas, obstacles, or room boundaries. The Roomba will report any found obstacles to the base station. The pathfinding algorithm will adjust and execute an appropriate path.	UNTESTED	Team

Questions?

Thank you!