Software Overview

Year: 2024 **Semester: Spring**
Team: #2 **Project: M.O.U.S.E**
Creation Date: 1/23/24 **Last Modified:** January 25, 2024
Author: Matthew Ghera **Email: mghera@purdue.edu**

1.0 Software Overview

The following list breaks down the primary software functionality of M.O.U.S.E:

- Read in sonar sensor data
- Web socket requests between microcontroller and web server
    - Send continuous stream requests to web socket from microcontroller
    - Receive, read, and process accordingly the response from the web socket requests
- Processing diagnostic LED output
- Read in battery life
- Control motor power appropriately (speed & direction)
- Transition robot between manual movement mode, movement scanning mode, movement record mode, and movement replay mode

1.1 Read in sonar sensor data

M.O.U.S.E will contain 4 sonar sensor (front, back, left, right) in order to detect movement. The software needs to be able to take in the data from all 4 sensors nearly simultaneously. The microcontroller will receive data from each sensor via the I2C communication protocol. The frequency the data will transmit at will be 15Hz as that is the max frequency of the sonar sensors we plan on using for long range [1]. Each communication will first start with a read request sent from the microcontroller to the sonar sensor. Then another request will be sent from the microcontroller to the sonar sensor where the sonar sensor will send back the distance data which will be the size of 2 bytes. This maximum frequency was chosen because I2C only allows for one peripheral to communicate at a given time. Due to this limitation, each sensor communication will need to be very quick, so that the microcontroller can receive data from each sonar sensor as close to simultaneously as possible. Each sensor will collect 20 samples of distance data in order to track if the distance changed over time. These 20 samples will be collected in four separate arrays which will be in an outer array, forming a 2D array.

1.2 Web socket requests between microcontroller and web server

The form of wireless communication that M.O.U.S.E utilizes will be through Wi-Fi. Through the Wi-Fi, the microcontroller will be able to send requests to an AWS server via a web socket using TCP. The embedded microcontroller will need to be able to stay connected to the web socket for a long amount of time (12+ hours), so there needs to be little to no disconnections and every disconnection needs to reconnect in a small window of time to prevent data loss. During this time, it will need the capability to make frequent requests (one every ~1 second). Additionally, it will need to be able to receive, read, and process the responses from the requests accordingly so that the correct software action can be enacted after each request response is received. These

messages will be made as small as possible as there is a low throughput threshold for the microcontroller.

1.3 Processing diagnostic LED output

Onboard the M.O.U.S.E will be 24 diagnostic LEDs to display digital values, such as the current battery life of the robot. Each LED will need to be controlled individually. To do this, a shift register daisy chain will be used which includes 3 8-bit serial-in parallel-out shift registers in total. The software will interact with this chain of shift registers by setting the bits and shifting in the respective amount for each LED using the microcontroller's internal clock to shift each bit and then sending a signal to another output pin that will act as the latch.

1.4 Read in battery life

Since battery power is such an important factor in a device like this, the software will also need to be able to read in the battery life as an input and be able to transfer that via the web socket and store it locally to display on the diagnostic LEDs.

1.5 Control motor power appropriately (speed & direction)

The wheel motors will be controlled via the user interface and web socket but also from the sonar sensor data. The control will need to be both quick and accurate so that there is minimal latency and so the robot's location can be accurately recorded relative to its starting position. Each motor will be controlled through PWM.

1.6 Transition robot between manual movement mode, movement scanning mode, movement record mode, and movement replay mode

M.O.U.S.E will have several different modes that it can be in.

- Manual movement mode where it is sending requests and receiving instructions from the web socket to move forward, backwards, left, or right
- Movement scanning mode where it is receiving and processing the sonar sensors input data and sending the processed data through the web socket
- The final mode is movement replay mode where it retraces the recorded manual mode's path

A quick note is that the movement replay mode is a stretch functionality for this project; nonetheless, all the infrastructure for storing the mode state will still be built and not changed whether the last mode is fully implemented or not.

2.0 Description of Algorithms

2.1 Shift register control

In order to control all 24 of the diagnostic LEDs individually through the 3 8-bit serial-in parallel-out shift register daisy chain, each LED will need to be addressed by shifting bits in respectively. In more detail, here is the general algorithm that will be used to do this:

1. Get the bit of each LED, for example, if trying to control LED 5 it would be 10000
2. Each bit will be shifted out as a pulse to the data pin, so sticking with the example of 5, it would shift out a 1 and then 4 zeros

3. Finally, additional shifting would be done for the same number shifted right 8 bits and 16 bits to hit all 24 LEDs

2.2 Current battery life

In order to calculate the current battery life, a lookup table will be used. There will be an input voltage from the battery coming into the microcontroller on a GPIO pin. This input voltage will be read as the input to a lookup table. This lookup table will map the input voltage to a current battery life. For example, if the input voltage is 4.5v, this could map to 80% battery life.
Aside: the lookup table will be formed from experimentation on how long the battery lives and the input voltage at each time point.

2.3 Sonar sensor I2C requests

Each sonar sensor will have a different address that is unknown, so first, a polling algorithm needs to be put in place that sends an I2C message to every possible address and checks to see if it gets back an acknowledgement. Once all four addresses have been found and recorded. The sensors need to be sent an I2C request to begin scanning a distance. Finally, after each sensor has returned success on scanning, one final request can be sent to each sensor to retrieve the reading and record the incoming data. This process will continue with a repetition of sending a request to read and a request to retrieve the data for 20 times in order to scan for movement.

3.0 Description of Data Structures

3.1 Sonar sensor data structures

As previously mentioned, the sonar sensors will use I2C, so one data structure correlating to the sonar sensor data is the incoming I2C data packet. Each distance read will use 2 packets and each individual I2C transaction will contain 1 byte of data (along with other metadata, such as the acknowledge bit) [1]. Both distance data bytes will be stored in the next data structure correlating to the sonar sensor. All of the I2C data packets will be stored in a 2D array. Each of the four sonar sensors will have an array within an outer array. Within each sonar sensor array will be a history of the last 20 most recent distance readings. This will allow for an array search to determine if the distance drastically changes on any given sensor which will correspond to a detected movement.

3.2 Motor data structure

Both the speed and direction of the wheel motor will be retrieved from a web socket request. This data will be stored in a motor struct, and each motor will use an instance of this struct. Then, when using PWM to control the motors, the motor structs will be read from so that the correct value is written out.

3.3 Battery life data structure

As previously mentioned, in order to get a current battery life from an input battery voltage, a lookup table will be used. This lookup table will be a hash table that maps each input voltage to the nearest tenth to a battery level to the nearest percent.

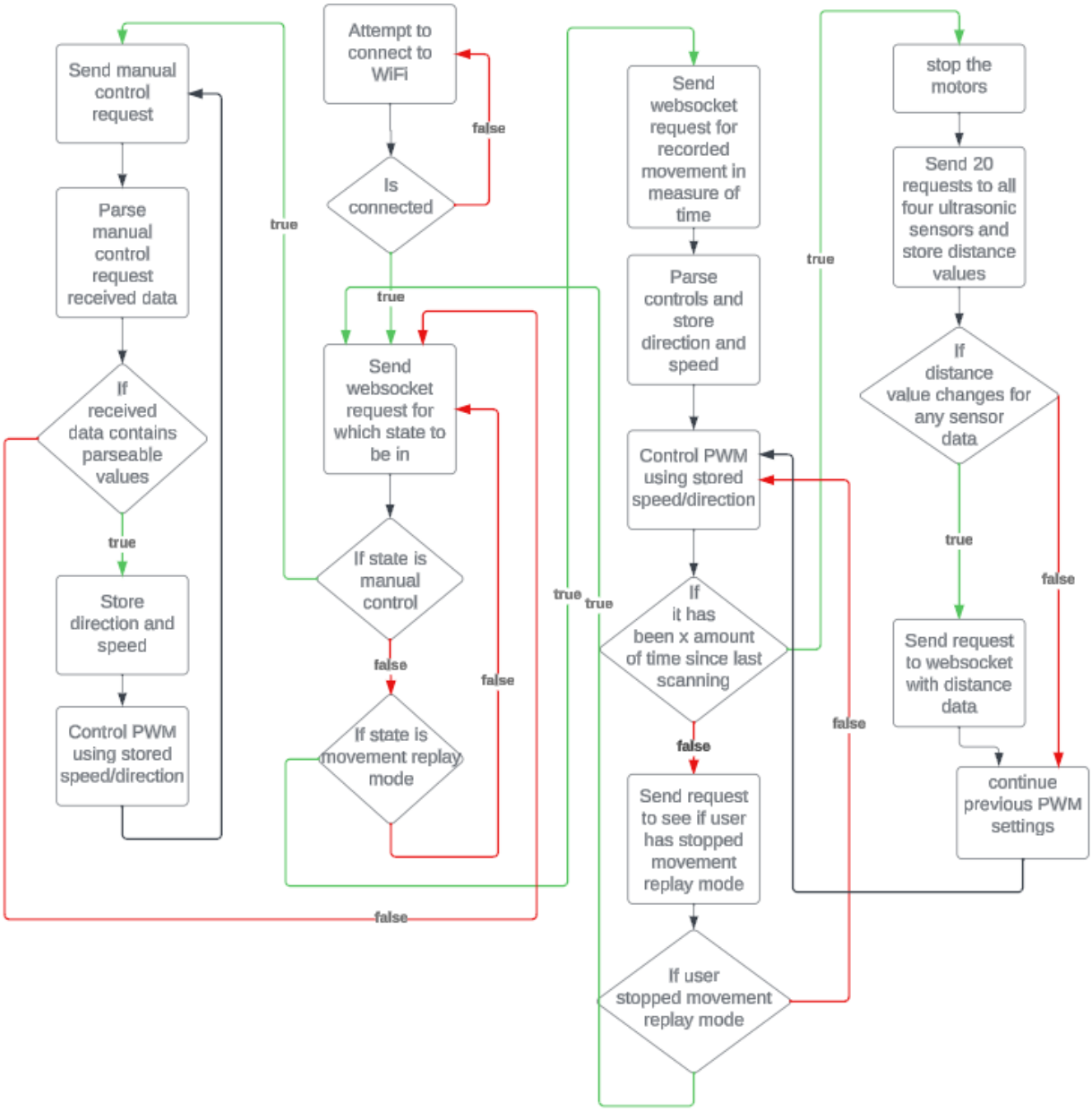
3.4 Web socket data structure

Each web socket request and response are important data structures that will enable transmission of data between the microcontroller and the web server. These will be in the form of a string. The data string coming from the microcontroller to the webserver will always be 2-3 characters to allow for the least amount of transmission latency as possible due to a low throughput threshold constraint. However, the data received from the webserver to the microcontroller will always be in the form of a key and a value as this does not contribute to the throughput threshold constraint.

4.0 Sources Cited:

[1] “I2CXL-MaxSonar-EZ Datasheet,” *MaxBotix*. https://maxbotix.com/pages/i2cxl-maxsonar-ez-datasheet (accessed Jan. 25, 2024).

Appendix 1: Program Flowcharts



Appendix 2: State Machine Diagrams