Software design description – Hardware

System overview:

\*A general overview of how the whole system should work.

The Mower is a vehicle constructed with an Arduino Mega2560 and a Raspberry Pi zero w. The Arduino controls the mower's sensors and motors, while the Raspberry Pi handle communication between the mower, frontend, and backend.

The Mower has two separate modes, manual and autonomous. When in manual mode the mower takes commands from a device connected with Bluetooth. When in autonomous mode the mower contains itself within a specified area while also avoiding front-end collisions.

System architecture:

\*A summary of how the arduino and raspberry pi are setup and communicating with each other. Should get a gist of how the hardware system is setup.

The Arduino and Raspberry pi is connected through serial UART communication with the use of a USB-B to USB-A cable which in turn is connected to a USB-B to microUSB converter. When in manual mode the raspberry pi sends commands to the Arduino which in turn controls the motors thereafter. When in autonomous mode the Arduino runs a state-machine that sends updates to the Raspberry pi every time a state is changed, this way the Raspberry pi can correctly calculate the position and send the appropriate information to the backend.

Detailed system design:

**MBot**

**Variables:**

**Functions:**

void \_delay(float seconds):

Input:

Output: None

Description: The program is paused for the time (in milliseconds) specified as parameter.

void move(moveDirection direction, int speed):

Input:

Output: None

Description:

void moveForward():

Input:

Output: None

Description: When this function is called, the mower will drive forward

void moveBackwards():

Input:

Output: None

Description: Call moveBackwards() to give the Mower more room to rotate.

void turnLeft():

Input:

Output: None

Description: When this function is called, the mower will turn left.

void turnRight():

Input:

Output: None

Description: When this function is called, the mower will turn right.

void stopMotors():

Input:

Output: None

Description: Call stopMotors() to prevent the Mower from crossing the line.

void collision():Not used?

Input:

Output: None

Description: when the Mower hits an obstacle, back then turn right.

void autoTurn():

Input:

Output: None

Description:

void isr\_process\_motorLeft(void):

Input:

Output: None

Description:

void isr\_process\_motorRight(void):

Input:

Output: None

Description:

int checkSensors():

Input:

Output:

Description: This function will check the state of the sensor.

String getOrientation():

Input:

Output:

Description:

int autonomousDriving(int currentState):

Input:

Output:

Description: When this function is called, the Mower will run on its own. The Mower will check the state of linesensorStateGlobal and the ultrasonic sensor in this function.

void bluetoothDriving(char nextState):

Input:

Output: None

Description: When this function is called, the mower will be controlled by the application.

void updateState(int data)Not used? Is this “bluetoothDriving”?

Input:

Output: None

Description: Will check the data given from Bluetooth and let the Mower be in different states by changing the state of mower StateGlobal to forward, backwards, left, right or stop(idle).

void setup():

Input:

Output: None

Description:

void \_loop():

Input:

Output: None

Description:

void loop():

Input:

Output: None

Description:

**Raspberry pi (Python)**

**Variables:**

* running – A Boolean flag keeping track of if the program should continue running.
* mode – A String varying between “Manual” and “Automated” which controls which mode the mower is currently in and what commands it should be executing.
* reversing – A Boolean flag keeping track of if the mower is reversing while in manual mode.
* turning – A Boolean flag keeping track of if the mower is turning while in manual mode.

**Functions:**

sendPositionRequest(x, y, sessionID, state, collisionFlag):

Input: x – The x-coordinate for the Mowers current position as an integer.

y – The y-coordinate for the Mowers current position as an integer.

sessionID – The ID of the current session as an integer.

state - defines if the session just started (“START”), is running(“RUNNING”), or has ended(“STOP”).

collisionFlag – A flag that should be set to True if a collision has occurred.

Output: None

Description: Function used to send a POST request regarding the mowers current position to the backend.

sendImageRequest(x,y):

Input: x – The x-coordinate for the Mowers current position as an integer.

y – The y-coordinate for the Mowers current position as an integer.

Output: None

Description: Function used to send a POST request with the most recently taken picture to the backend.

bluetoothInit():

Input: None

Output: client\_sock – The client socket connected to the rpi.

Description: Function used to initiate the Bluetooth server socket and awaits a connection from a client.

class CalculatePosition:

\_\_init\_\_(self):

Input: None

Output: None

Description: Initiates a CalculatePosition object. This class is used to be able to easily run and pause a separate thread that calculates the mowers current position through dead reckoning and trigonometry.

terminate(self):

Input: None

Output: None

Description: Function called to pause positional calculation in the separate thread. Simply sets a flag until next call to run.

run(self, speed, newDirection):

Input: speed – Defines the speed the mower is moving at.

newDirection – Defines the current direction the mower is moving in, should be an integer within the range of 0-360.

Output: None

Description: Function called to begin or continue calculating the mowers position in the separate thread.

class ReceiveBluetooth:

\_\_init\_\_(self, client):

Input: client – the socket client connected to the rpi.

Output: None

Description – Initiates a ReceiveBluetooth object. This class is used to be able to receive messages from the client socket from a separate thread running the background.

terminate(self):

Input: None

Output: None

Description: Function called to pause receiving Bluetooth messages in the separate thread. Simply sets a flag until next call to run.

run(self):

Input: None

Output: None

Description: Function called to begin receiving Bluetooth messages. When a message is received, it is stored in a variable and a flag is set to tell the main-thread that there is a new command. When the command has been handled by the main thread, the class can continue looking for new messages.