Robotic Waiter for Restaurants

Group 26

Jun Pun Wong, Kausik Venkat, Cheng Jin

Introduction

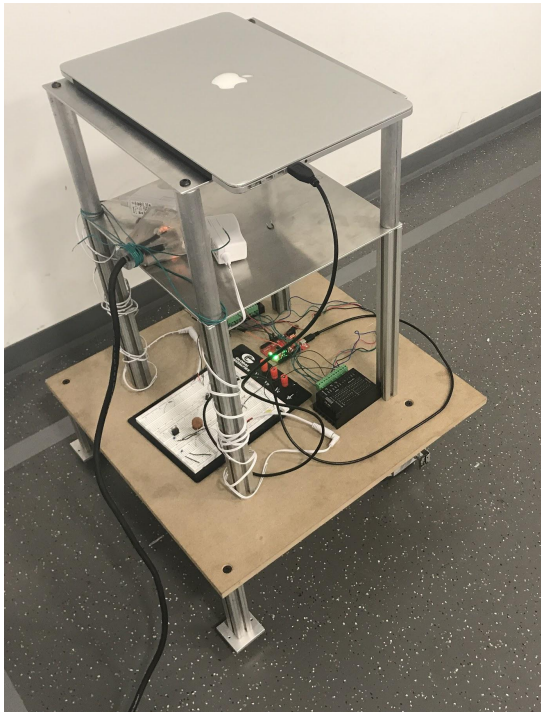
- Manpower is an issue in restaurant industry
- Workers can be inefficient, no-shows
- Need to be trained and paid as well
- Robotic waiter to assist restaurant staff in delivering food from kitchen
- Robotic waiters would be cheaper in long-term

Objective

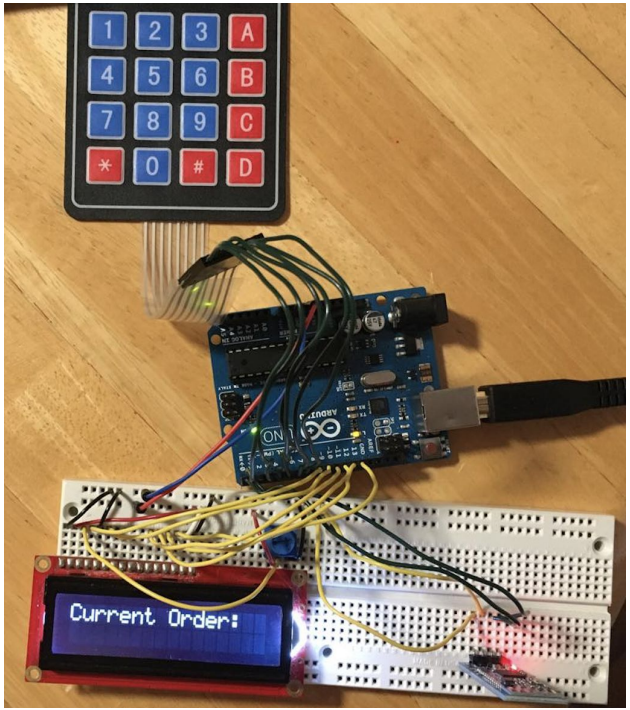
- Kitchen microcontroller (MCU) would send robot table information to robot MCU
- Robot MCU communicate with Raspberry pi to get the path and control motor drivers to navigate robot
- Kitchen MCU would display table being served by the robot
- Table locations pre-programmed

Robotic Waiter

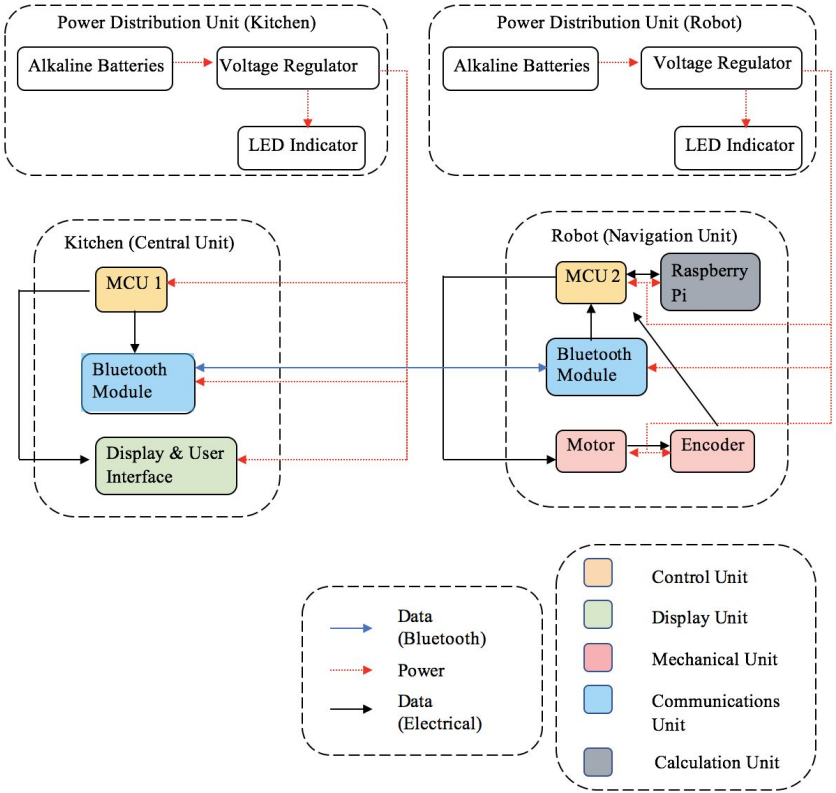
Robot (Navigation Unit)



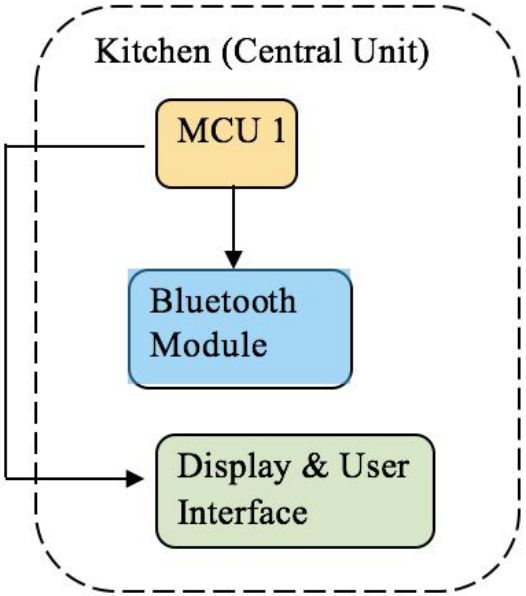
Kitchen Unit (Central Unit)

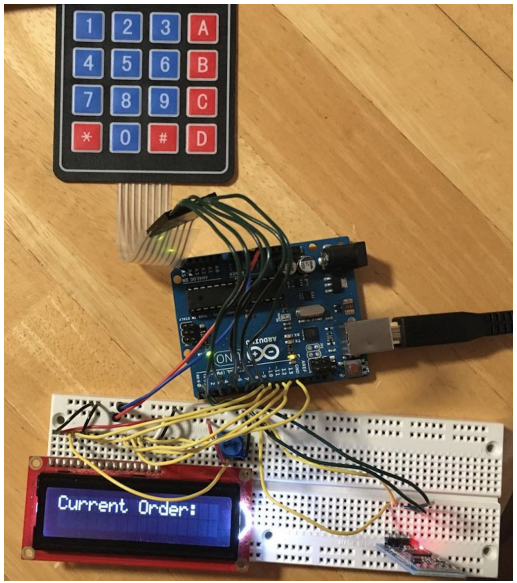


Design Block Diagram

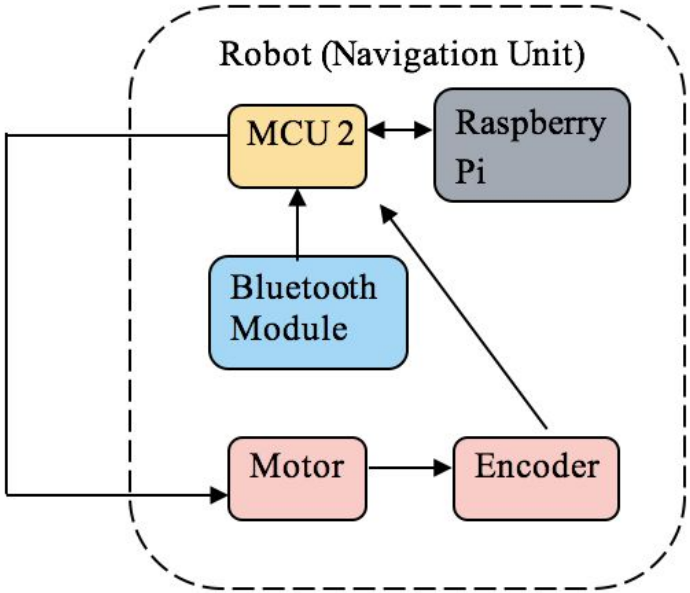


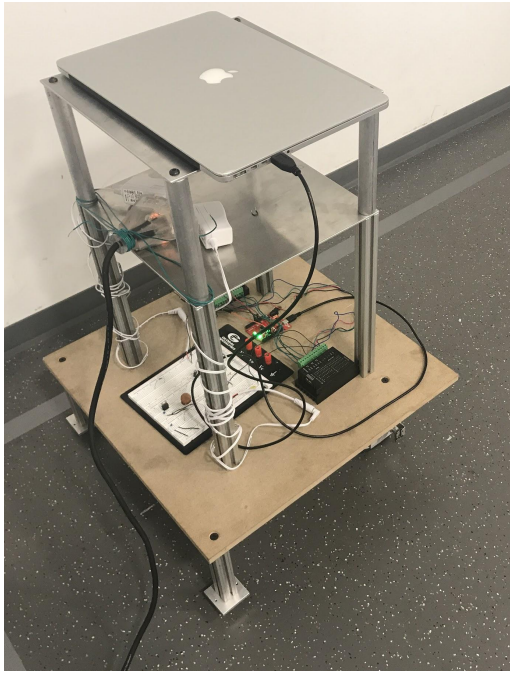
Kitchen Unit (Central Unit)





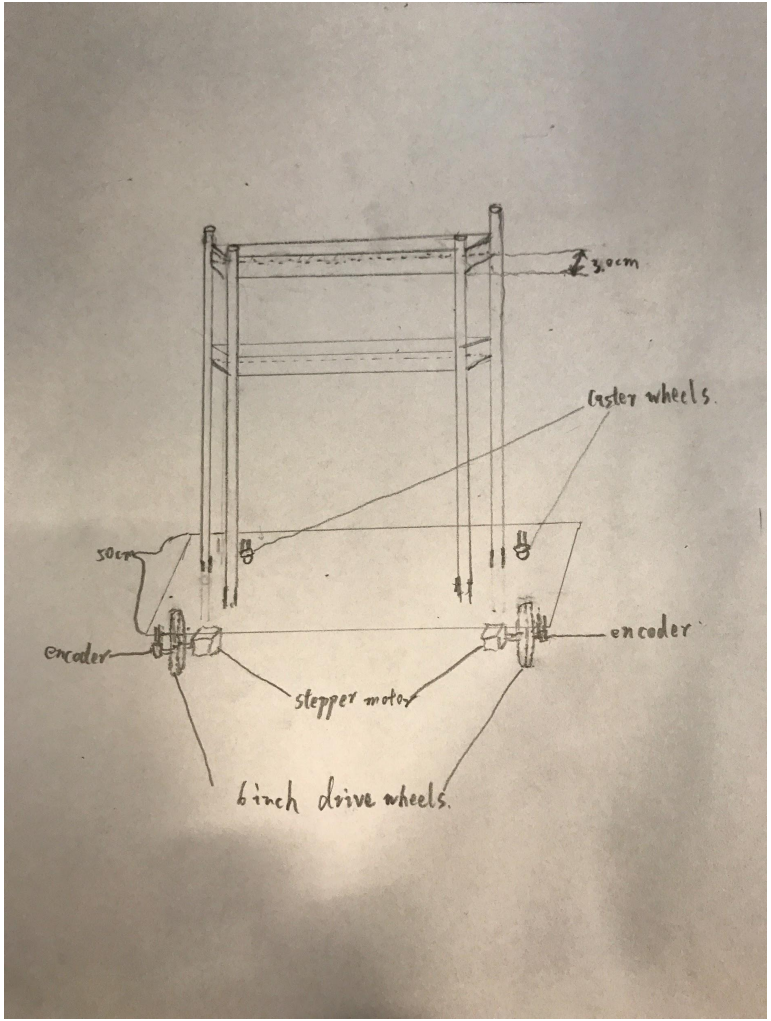
Robot Unit (Navigation Unit)

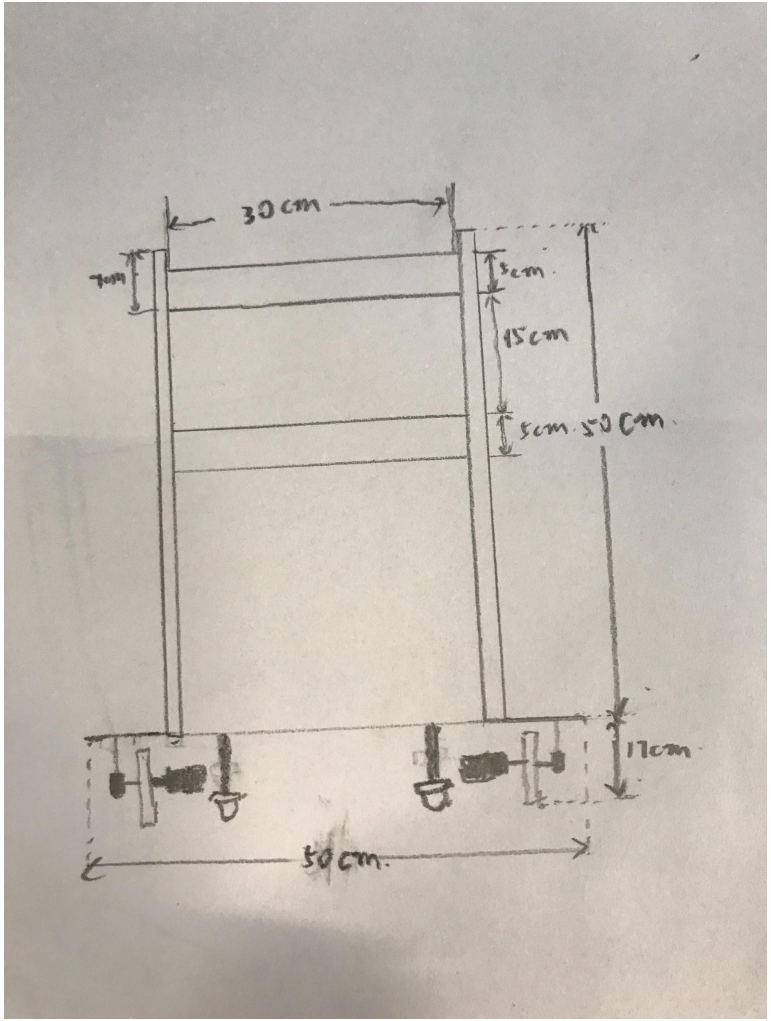


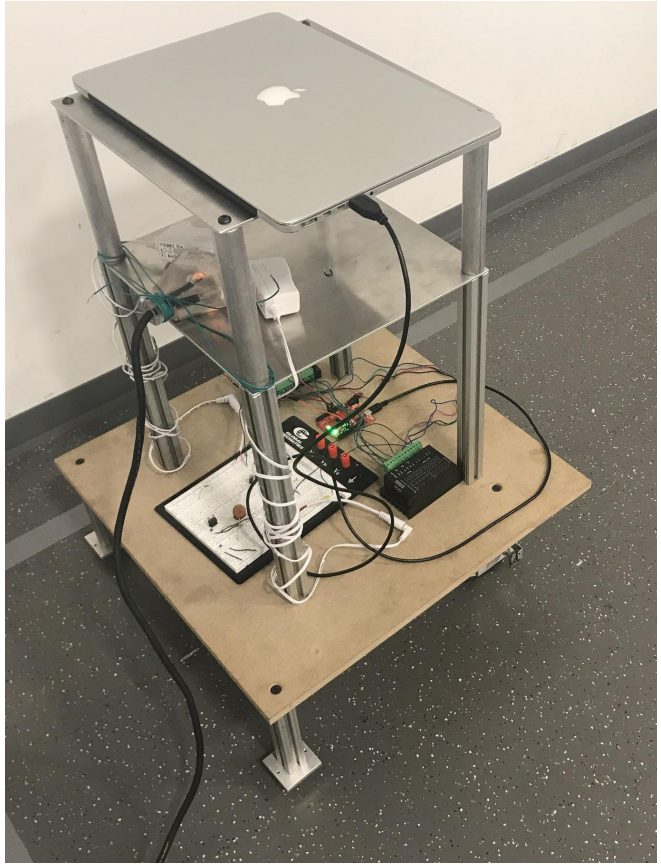


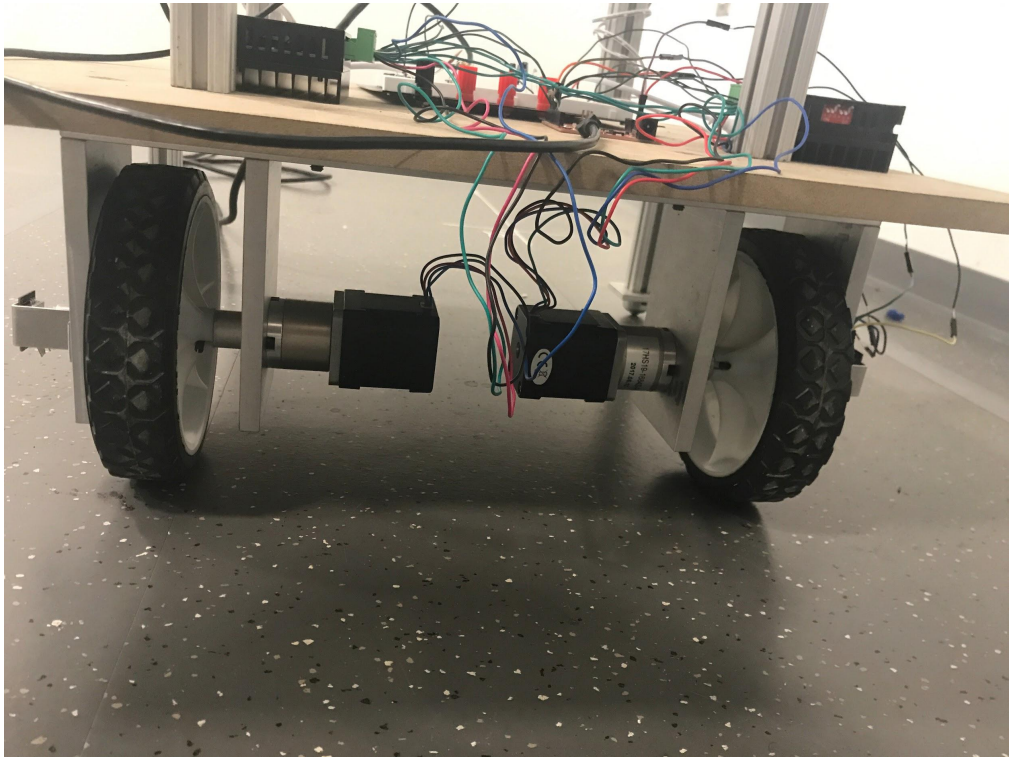
Physical Design Components

1. Overall Structure
2. Stepper Motor
3. Motor Driver
4. Encoder


30cm
caster wheels
50cm
encoder
encoder
stepper motor
6inch drive wheels

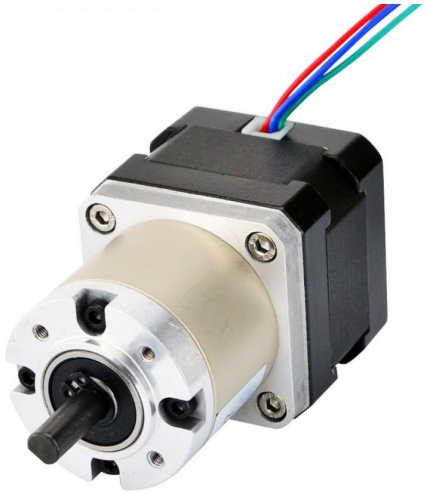

30 cm
7cm
5cm
15cm
5cm
50 cm
17cm
50 cm.





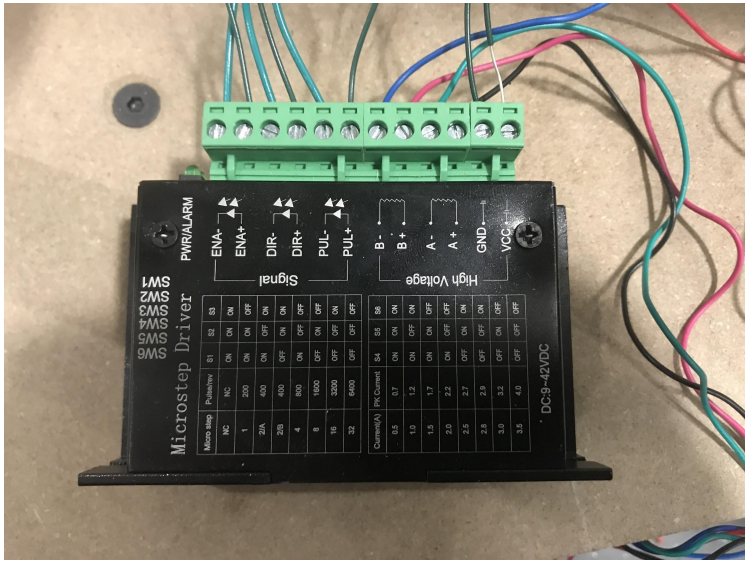
Stepper Motor Details

1. Nema 17 bipolar stepper motor
2. Gear ratio: 27:1
3. Step angle 1.8 degree
4. Backlash ≤ 1 degree
5. Voltage ≥ 12 V
6. Current ≥ 1.7A
7. Balance between torque and speed.



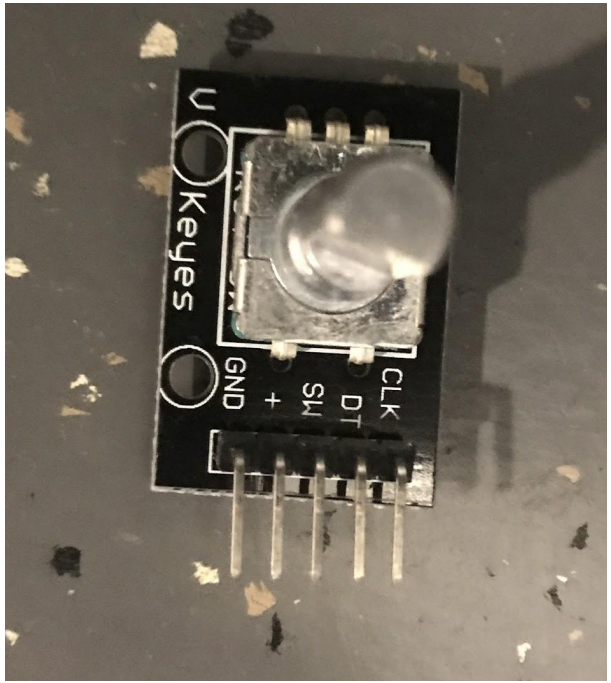
Motor Driver Details

1. Voltage range 12V-48V
2. Maximum Current 5A
3. DIP Pin setting
4. Signal Pins: PUL+, PUL-, DIR+, DIR-



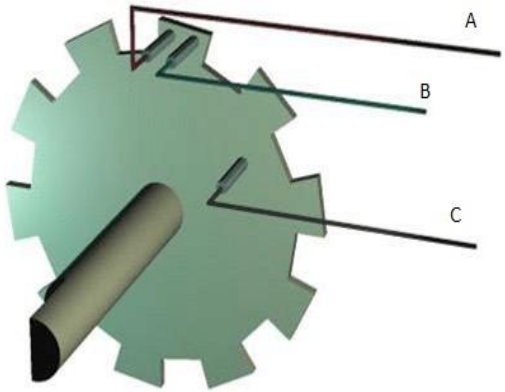
Encoder Details

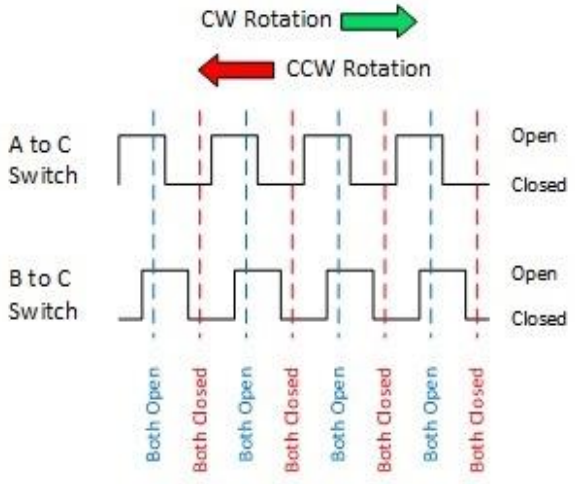
1. Pins
2. One rotation is 30 encoder count
3. Room is represented in unit of encoder count
4. Theory and control



Encoder control

1.Reads the "current" state of the outputA
2.If the previous and the current state of the outputA are different, that means a Pulse has occured
3.If the outputB state is different to the outputA state, that means the encoder is rotating. We increase the encoder count.



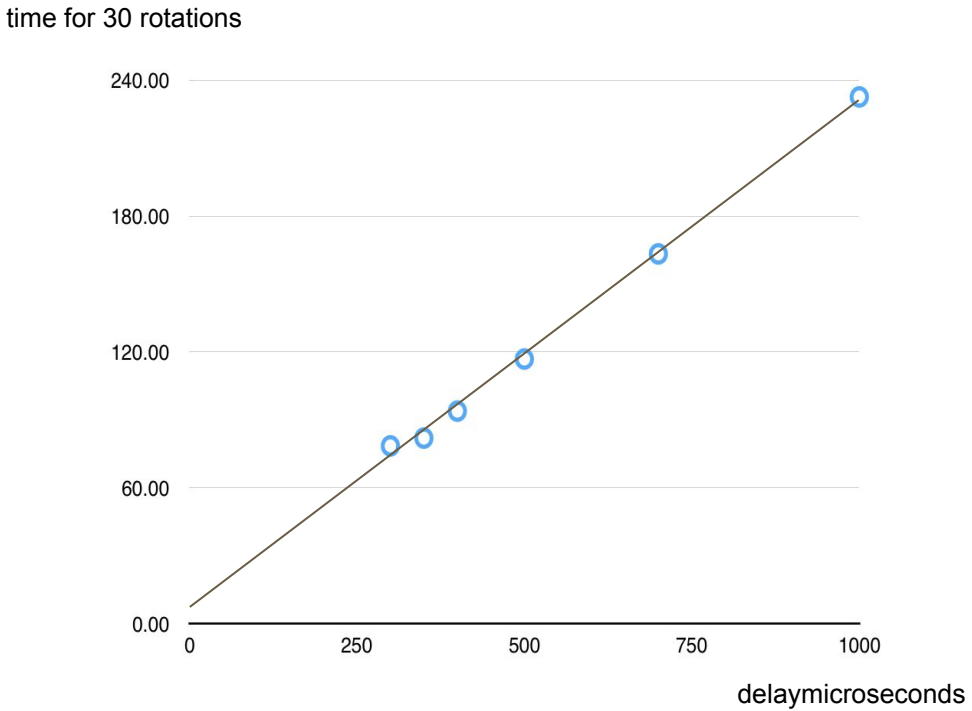


Movement & Calibration Methods

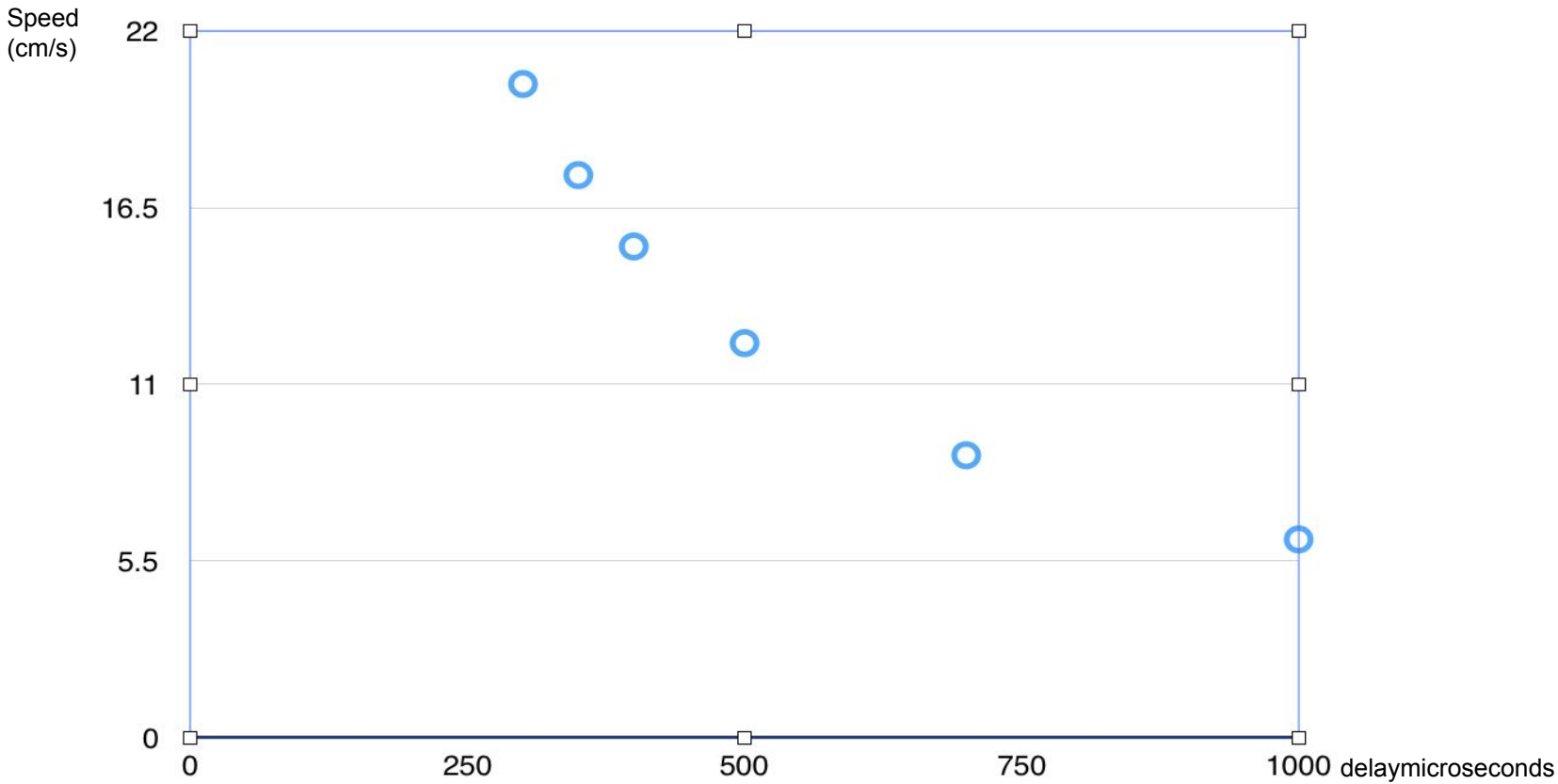
1. Straight line
2. Turning

Straight Line Calibration

1. Step pin on the driver can control motor when to move or stop
2. Dir Pin can control motor's direction (symmetry, both High or Low will make robot turn)
3. Changing delay time between pulse can change the speed.

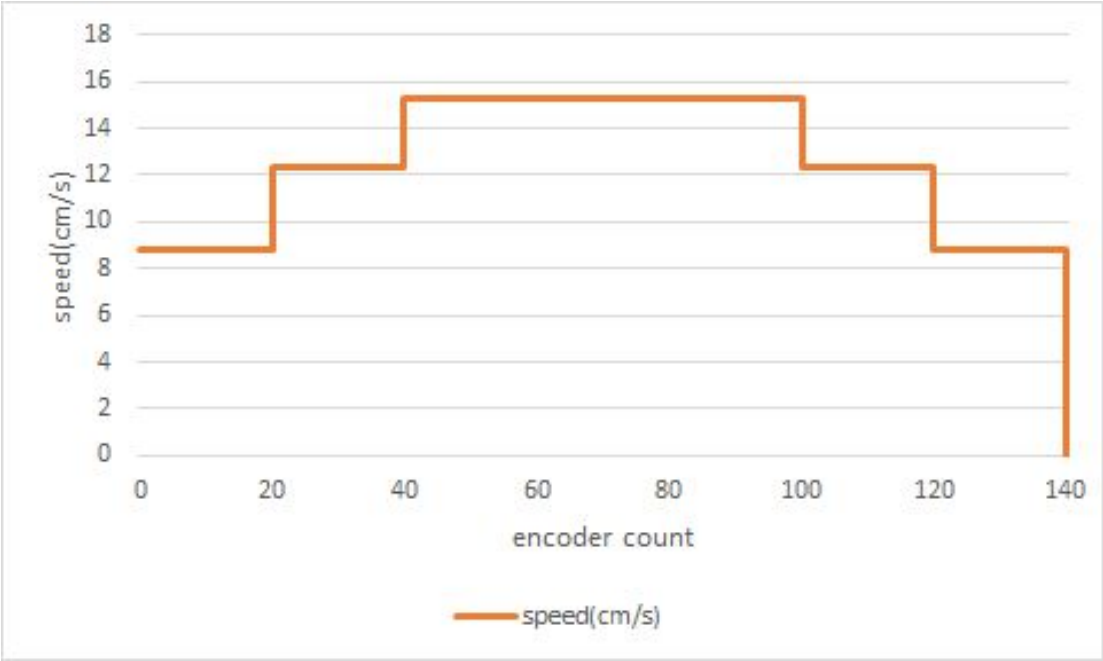


Speed vs delay time



Acceleration

Step function



Requirement for straight line movement

Error in displacement in x-axis should be less than 5 cm for every 1 meter moved in y-axis.

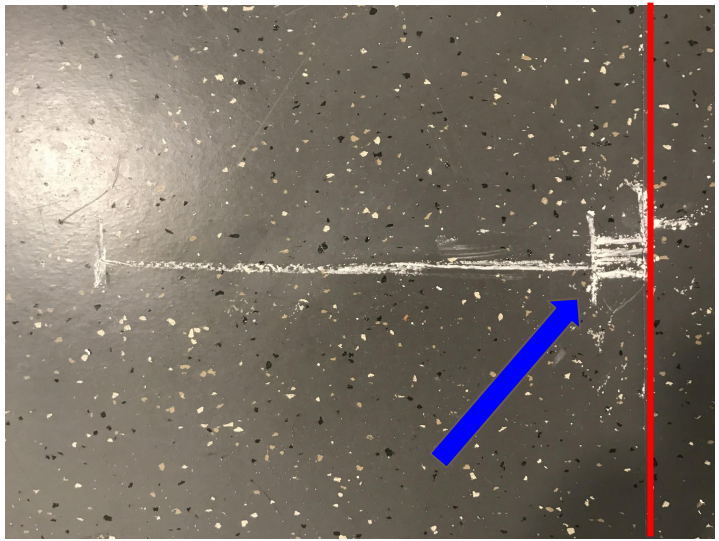
Error in movement in y-axis should be less than ±0.8cm for every 1 meter moved

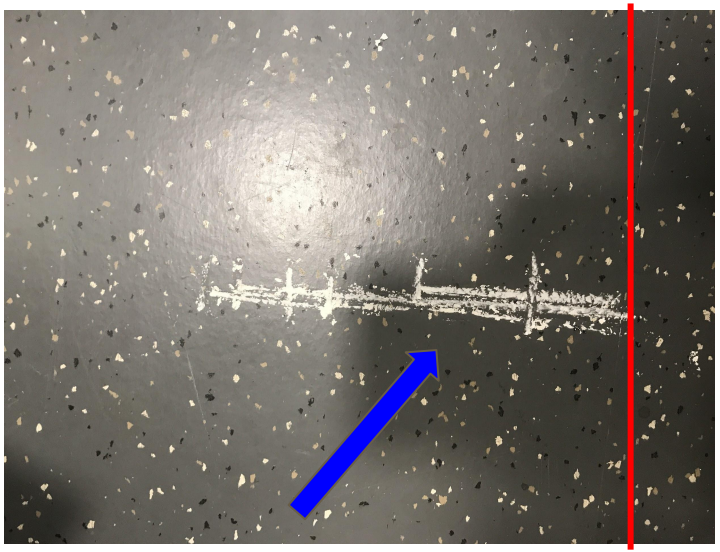
5. Calibration

(a) Reason: Influenced by floor, resistance, Encoder error(y direction), wheels and motor(x direction)

(b) Y direction error Vibrates from -0.4 to 0.8 cm for 200 encoder count(319.2cm)

(c) X direction error 2.5 inches(6.35cm), 4.25 inches(10.8cm)



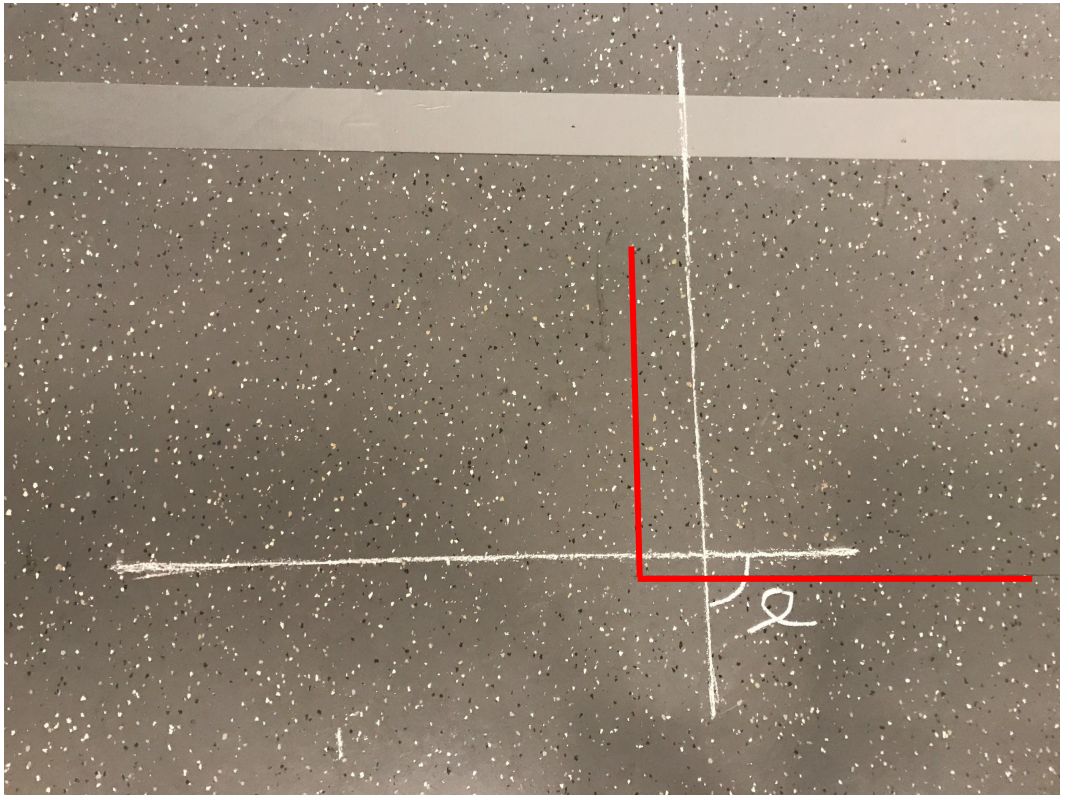


Turning Calibration

1. DIR pin
2. Using encoder count to control (when encoder count = 16)
3. Using arduino to control

4. Turning code and calibration

Error in angle when turning should be within ±5°



Power Distribution Unit Design

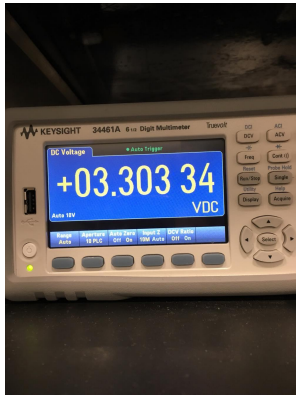
1.) Motor driver requires at least 20V (with at least 1.7A)

2.) ATMEGA328P operating voltage is 1.8V - 5.5V

3.) HM-10 Bluetooth Module operating voltage is 3.3V (50mA)

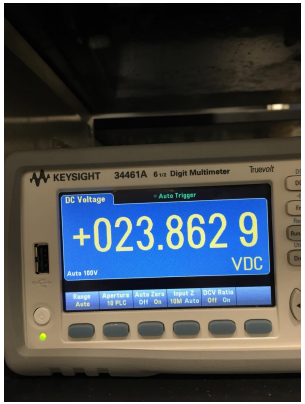
4.) Raspberry Pi requires 5V (2A)

Power Distribution Unit Verification



Output Voltage from 3.3V regulator

Output Voltage from 5V regulator



Output Voltage from 24V regulator

Power Distribution Unit Calculations

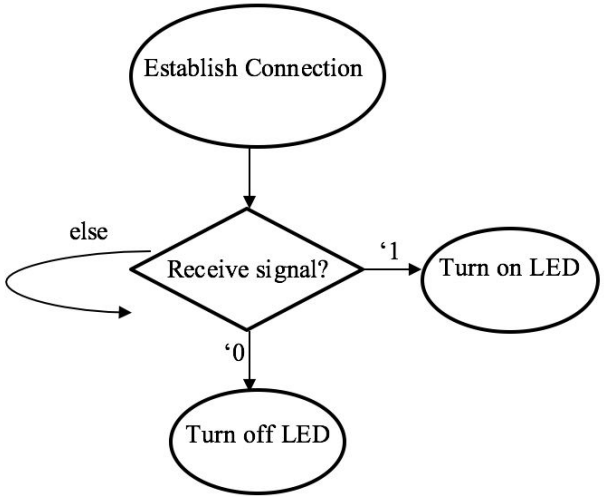
$$Battery\ Life\ (hour) = \frac{Battery\ Capacity\ (mAh)}{Load\ Current\ (mA)} \times 0.7$$

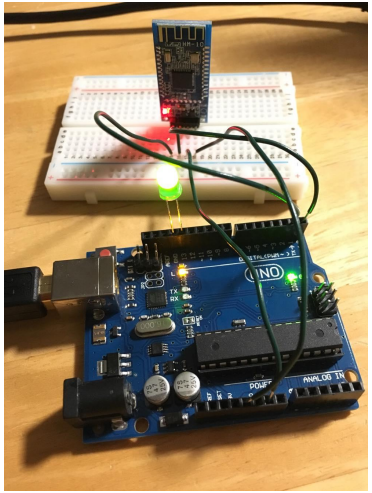
*0.7 is a constant to account for external factors

Drawing 1.7A using the 24V Voltage Regulator,

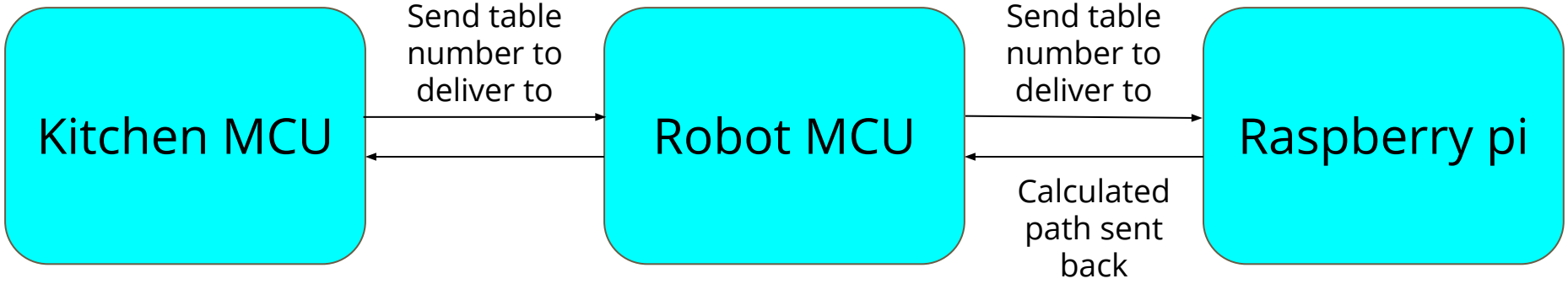
we obtain that the battery life to be 0.23 hours (13.8 minutes)

Bluetooth Communication Verification





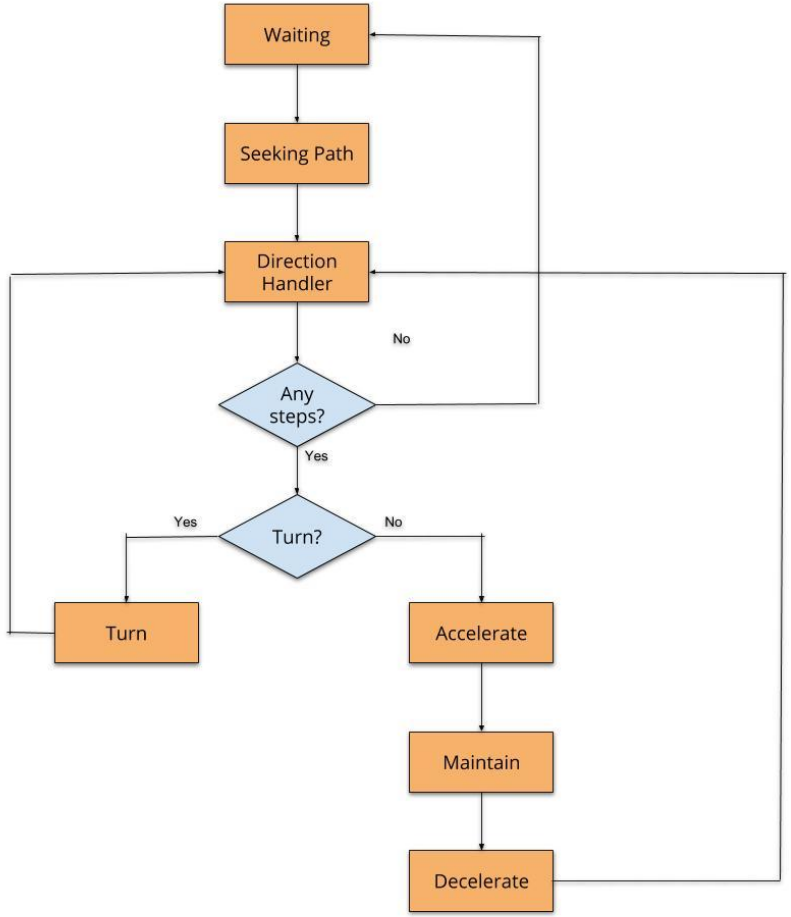
LED is turned on by BLE module


Kitchen MCU
Send table number to deliver to
Robot MCU
Send table number to deliver to
Raspberry pi
Calculated path sent back

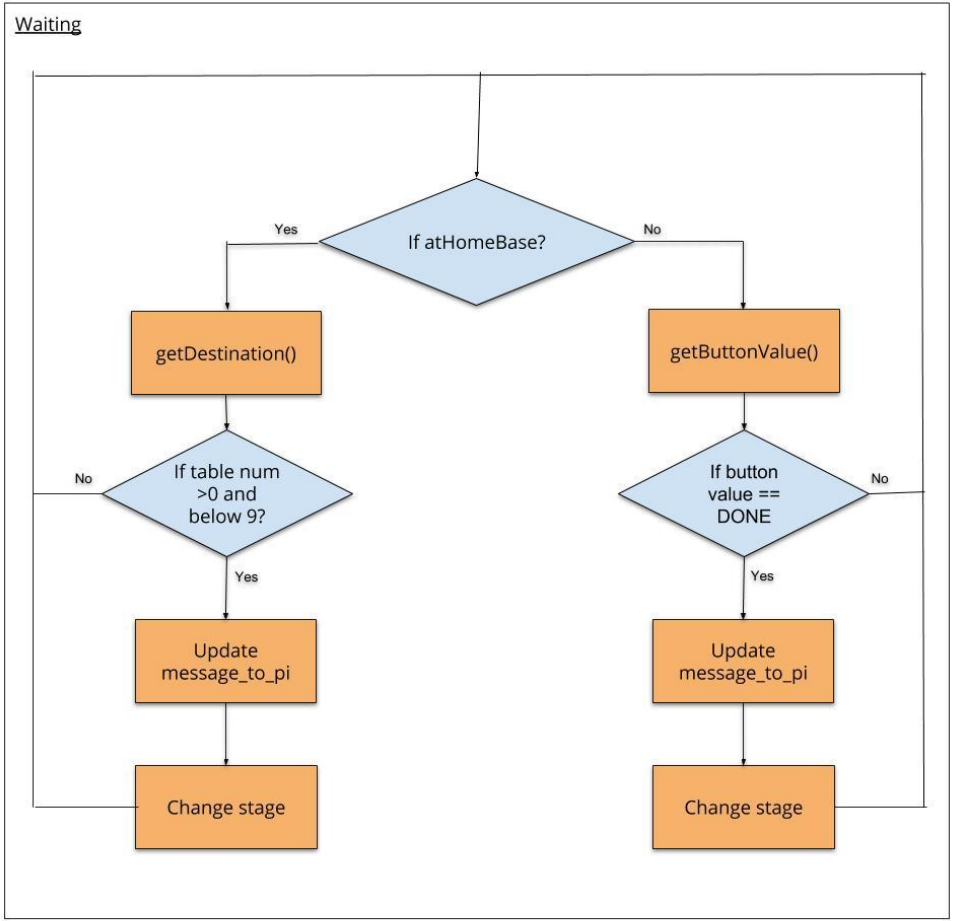
Path naming methodology

- Message from Raspberry pi will be interpreted as a string
- Alternative sequence of letters and numbers
- If letter == ‘S’, moveForward. If letter == ‘T’, turn.
- Number that follows the letter is either the distance to moveForward or direction to Turn
- Letter == ‘D’ signifies end of path
- i.e. **S5T1S3T3D**

Robot's MCU software



WAITING


Waiting
If atHomeBase?
Yes
No
getDestination()
getButtonValue()
If table num >0 and below 9?
No
Yes
Update message_to_pi
Change stage
If button value == DONE
No
Yes
Update message_to_pi
Change stage

SEEKING PATH

```
If message_collected:
    message_collected = False
    stage = direction_handler
```

DIRECTION HANDLER

```
Extract next ‘step’
If next_step  == ‘D’:
        atHomebase = !atHomebase
        stage = WAITING
Else
        Get parameter for turn/moveForward
If next_step == ‘T’:
        Set turn_count
        Stage = TURN
Elif next_step == ‘S’:
        Set distances for accelerate, maintain, decelerate
        Stage = ACCELERATE
```

TURN

If turncount > 100:

    Turn for 100 counts // for loop
    Subtract 100 from turncount

Elif:

    Turn for turncount number of times
    Turncount = 0
    Stage = DIRECTION HANDLER

ACCELERATE/MAINTAIN/DECELERATE

```
moveForward(speed)
updateEncoderCounter()
If encoder_counter >= encoder_breakdown:
        Move onto next stage
```

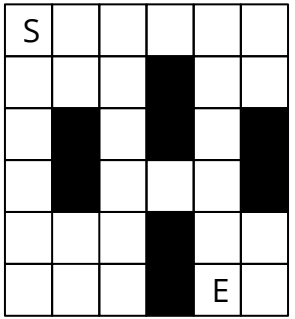
- Same psuedo code for all 3 stages
- Differences:
  - Speed to move forward at
  - Encoder_breakdown value (pre-calculated in DIRECTION HANDLER stage)
  - Next stage to go to

Arduino to Raspberry pi

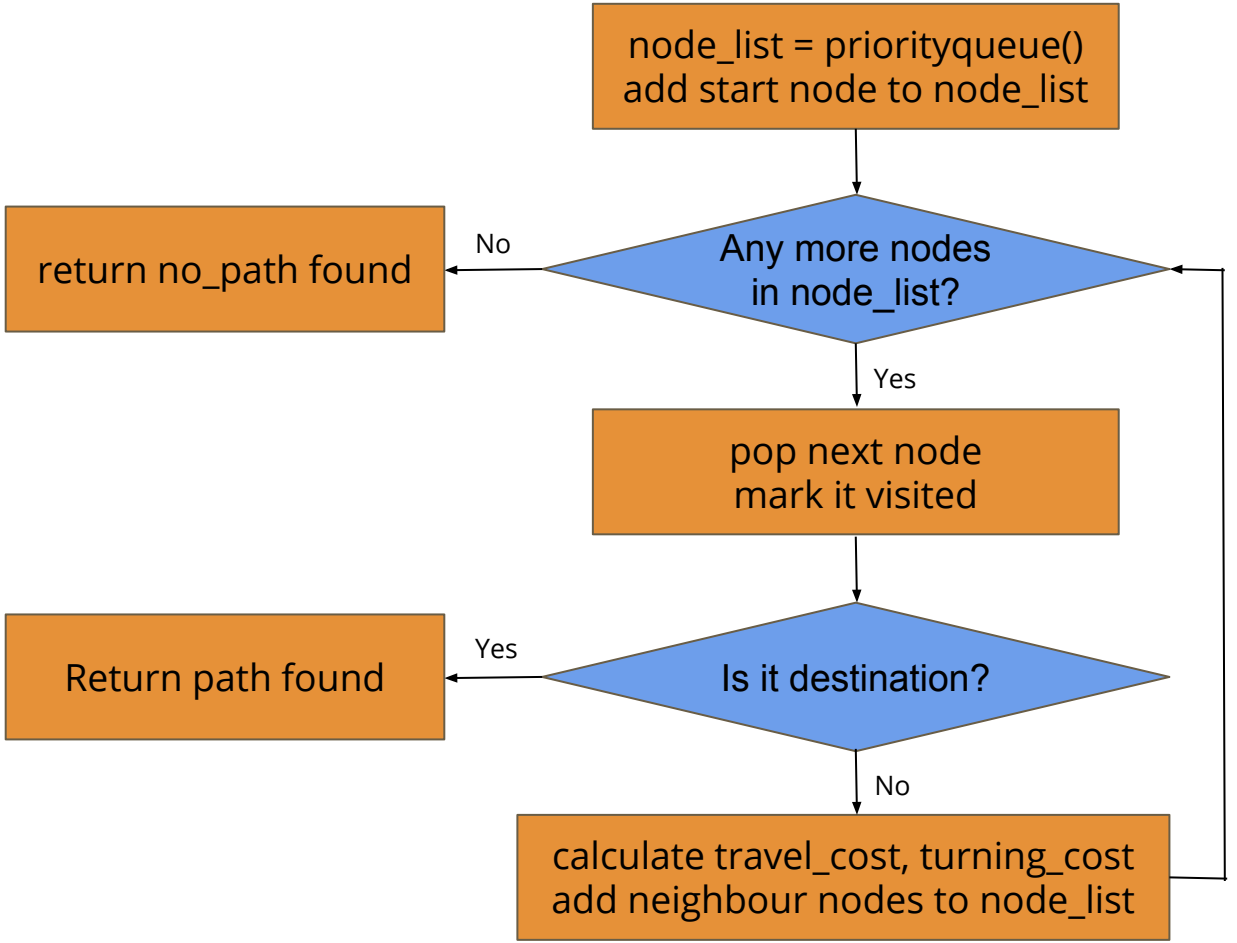
- i2c communication protocol
- Raspberry pi is master and Arduino is slave
- Connection initiated in setup()
- Data request every 5s

Path-finding algorithm

- Restaurant mapped to grid
- Astar search
- Heuristic function: travel_cost + turn_cost]
- Requirement: <5s



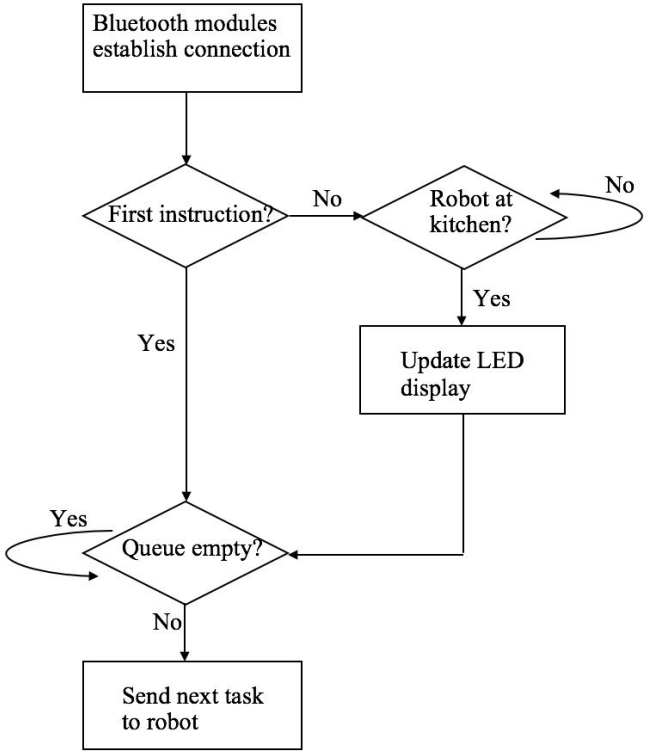
Path-finding algorithm



Testing

- Smaller functions such as moveForward, turn etc were tested modularly
- Entire cycle from WAITING to TURN/ACCELERATE-MAINTAIN-DECELERATE was simulated using fake path
- A-star search was tested on smaller mazes

Kitchen Unit (Central Unit) Implementation



What worked?

- Bluetooth communication between kitchen and robot
- Voltage regulators
- Motor drivers, encoders
- Arduino to Pi communication

What didn’t work?

- ATMEGA328P Chip could not be bootloaded
- Battery capacity was too low
- Robot movement was influenced by the floor. It gave unexpected errors when it moves which was not consistent.
- A-star path-finding algorithm did not work for full-lab room

Improvements

1. Integrate Path-Finding Algorithm with Robot Movement
2. Use rechargeable batteries with higher capacity
3. Use smaller Bluetooth communication modules

Future development

1. Implement a charging port using IR beacon
2. Implement collision detection & avoidance
3. Improve path-finding methodology such that tables need not be pre-programmed

Thank you!