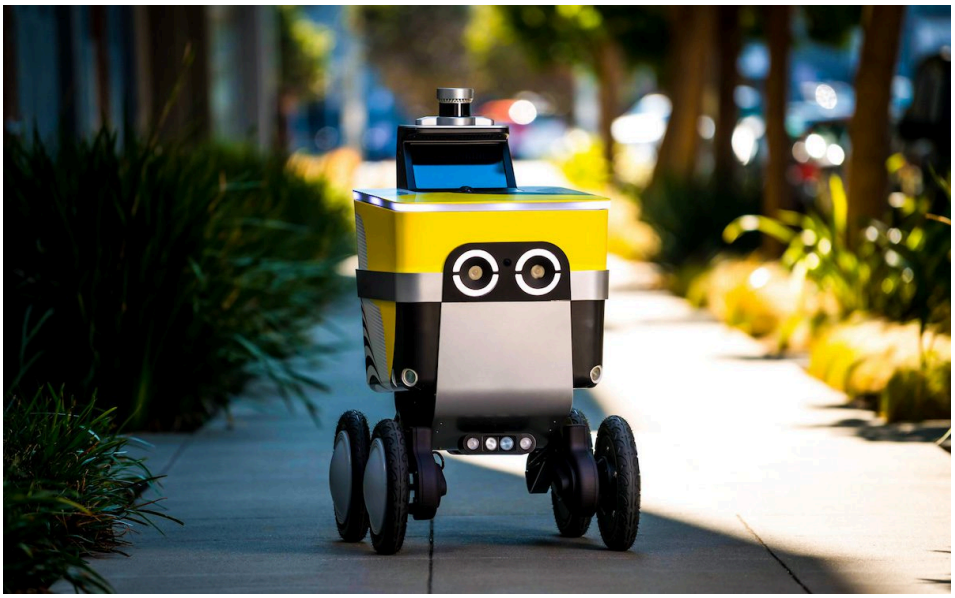
# G1 HOMEWORK REPORT
# EG2310 Group 9




| | |
|---|---|
| Aditi Joshi | A0265848L |
| Leow Kai Jie | A0254460J |
| Jing Ming Yuan | A0258733W |
| Rupan Kumar Manogar | A0259243B |

# Table of Contents

## 1. Problem Definition

Our aim is to design an ecosystem for a restaurant delivery robot that can successfully deliver a soda can in a room layout with 6 tables with customers and some obstacles on the path. Throughout the project, we will be designing two components for the mission: Payload dispenser and the delivery robot.

**We have divided our ideation into four major sections:**

| S.no | Robot Requirements | General Description | Specific Details |
|---|---|---|---|
| 1 | Autonomous Navigation | Helps the robot navigate around using a mapping device paired with some algorithms. | - Mapping Technique<br>- Navigation algorithms |
| 2 | Digital Interface between the turtlebot and dispenser | Establish a digital connection that allows communication between the two for smoother functioning. | - Microcontroller<br>- Near range communication<br>- Number pad<br>- Communication between microcontrollers |
| 3 | Physical Interface of the entire system | Proper alignment between both the components so that the payload can be placed well. Also provide signal to the bot that the can has been loaded. | - Mechanical function to move the can<br>- Weight sensor installed in the turtlebot and buzzer |
| 4 | Designs | Proper and feasible design of both components so that the delivery can take place | - Dispenser<br>- Modifications to the turtlebot |

## 2. Literature Review

### 2.1. Autonomous Navigation

#### 2.1.1. Mapping Technique

- Mapping is one of the crucial elements of autonomous navigation. The technology we will be utilising is the Light Detection And Ranging (LIDAR) together with the Simultaneous Localization and Mapping (SLAM) technique to form an accurate map.
- LIDAR creates a 2D representation of the environment by emitting light in the horizontal plane. Based on the captured reflected light, an accurate representation of the distance of the object is known.

#### 2.1.2. Navigation Algorithms

- An efficient algorithm is another essential component of navigation. Efficiency in this case will be measured in terms of the time it takes to move from the dispenser to the customers' table. Therefore, we will be employing shortest path algorithms. We have narrowed down the multitude of algorithms out there to just 3, Dijkstra Algorithm, A-star Algorithm and D-star Algorithm.

| | Dijkstra | A* | D* |
|---|---|---|---|
| Description | An algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. | It is considered as an extension of the Dijkstra algorithm, but tries to improve the runtime by using a heuristic to find the optimal solution | In contrast to A*, which follows the path from start to finish, D* begins by searching backwards from the goal node. This means that the algorithm is actually computing the A* optimal path for every possible start node |
| Time Complexity | Worst Case $O(n^2)$, n is number of nodes | Worst Case $O(b^n)$, b is branching factor and n is number of nodes | Greater than A*. **(See Appendix B)** |
| Ease of Implementation[1] **(Appendix B)** | Easy | Medium | Hard |
| Known Information[1] | Unknown map | Known Map | Partial Map |

- Dijkstra algorithm works on the basis that the map is unknown, resulting in an uninformed search where no information about the environment is known and a brute force method is employed, resulting in greater time complexity than A* or D*. Also known as heuristic searches, A* and D* algorithm takes in other parameters such as cost to goal and cost taken from source in a partially or fully known map to produce much lesser time complexity than Dijkstra's algorithm. A* works when the entire environment is known while D* is "capable of planning paths in unknown, partially known, and changing environments in an efficient, optimal, and complete manner" [2]
- At face value, D* looks like it is better than A* which is better than Dijkstra's algorithm, however the implementation of Dijkstra is easier as compared to A* which is easier than D*. This is because in A* and D*, we need to compare choose the best heuristic from heuristics such as Euclidean[3] distance or Manhattan[4] distance to correctly implement the algorithm

## 2.2. Digital Interface between the turtlebot and dispenser

### 2.2.1. Microcontroller

- It is inevitable that the dispenser needs a "brain" to receive incoming data such as the turtlebot being in place and then execute the dispensing while also sending table information to the turtlebot.

---

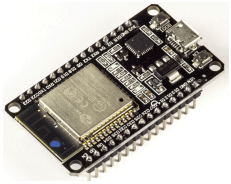
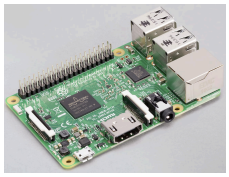
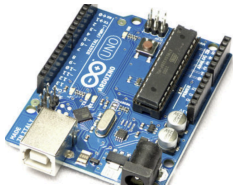
[1] https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf

[2] A. Stentz, "Optimal and efficient path planning for partially known environments," in Intelligent Unmanned Ground Vehicles. Springer, 1997, pp. 203–220.

[3] https://www.wiwi.uni-kl.de/bisor-orwiki/Heuristics:_A*-algorithm_1#:~:text=The%20A*%20(or%20A%20star,to%20find%20the%20optimal%20solution.

[4] http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

- Some functionalities that we were looking for include, programmable GPIO pins, wireless capabilities and low cost.
- Several boards that could perform these tasks caught our eye during our research, such as ESP32, Raspberry Pi and Arduino.

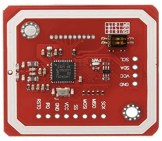
| | | |
|---|---|---|
|  |  |  |

### 2.2.2. Near Range Communication

- When the turtlebot is at the dispenser, it has to “inform” the dispenser that it is in position and ready for the next order. RFID and NFC are 2 possible near range communication technologies that we will be exploring for this purpose.

| | RFID | NFC |
|---|---|---|
| Description | Radio frequency identification system (RFID) is a contactless one-way communication method at varying distances. | Near Field Communication (NFC), allows for two-way communication and requires action by the user. |
| Range | Hundreds of feet | Very short, within 0.1 metres |
| Communication | Able to read large number of tags at once, 1-way communication | One tag at a time, 2-way communication |

- The MFRC–522 card reader and the PN-532 NFC-RFID Module are 2 suitable devices that we found.

| | PN532 | MFRC-522 |
|---|---|---|
| Image |  |  |
| Communication Interface | I2C, UART, SPI | I2C, UART, SPI |
| Range | Supports NFC with a 50mm range | 5 cm |
| Protocols | ISO 14443A, FeliCa, ISO 14443B, NFCIP-1 | ISO14443A |

### 2.2.3. Number Pad

When an order has been placed, the turtlebot needs to deliver it to the correct table. The number pad will be used to send the table number to the turtlebot.


### 2.2.4. Communication between microcontrollers

Information needs to be sent between the dispenser's microcontroller and the bot's controller in order for functions to be performed seamlessly. We are opting for 2 wireless options, Bluetooth and MQTT.

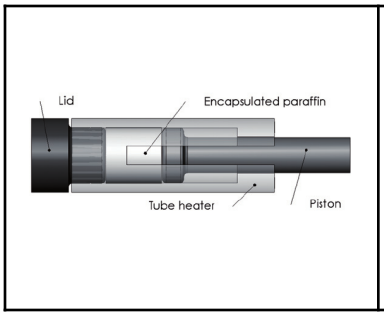
| | Bluetooth | MQTT |
|---|---|---|
| External requirements | Bluetooth module | Message Broker |
| Reliability | Designed to be resilient by using adaptive frequency hopping and sending data in fast, small packets [5] | Offers 3 Quality of Service (QoS) levels to choose from to ensure reliability of message being sent successfully |
| Range | More than a kilometre [6] | Internet Connection |

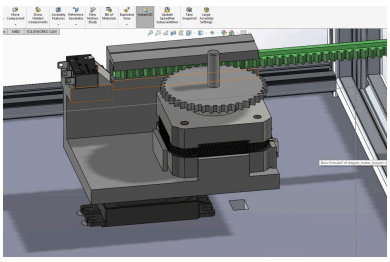
## 2.3. Physical interface of the entire system

### 2.3.1. Mechanical Functions (to move the can)

**Self-made Linear Actuator**

In order to move the soda can from the dispenser to the turtlebot, a linear actuator that extends and pushes any object in front of it can be used. By using the similar mechanism of the linear actuator, the selfmade linear actuator will be powered by a servo motor to rotate the gear and convert the angular motion to translational motion. To ensure the exact same distance of the linear actuator pushed, the number of rotations of the gear will be counted by the single board computer in the dispenser. Other supplementary design to ensure the proper working for the linear actuator will be the stopper at the end of track and side boundaries of the track to ensure the gear not sliding off from the track.

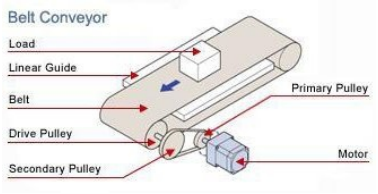


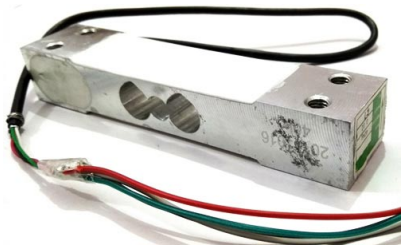
**Conveyor Belt**

Another way to move an object is through the use of a conveyor belt system. A motor drives a pulley with a belt to convert rotary motion to linear motion and move a load on its belt. Moreover, load weight sensors can be integrated in order to determine when the motor should start and stop.

[5]https://www.bluetooth.com/blog/bluetooth-range-and-reliability-myth-vs-fact/
[6]https://www.bluetooth.com/blog/bluetooth-range-and-reliability-myth-vs-fact/




### 2.3.2. Weighing Sensor



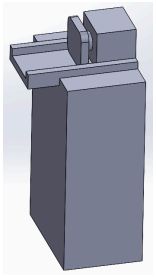
Although the dispenser can also provide signals to the robot after the payload is pushed from the dispenser to indicate that loading is complete, there is a possibility that the payload did not load fully into the bot's "holder" owing to inaccuracy from the actuator. A weight sensor module will be installed to double-check the loading procedure to make sure that the payload is correctly delivered to the holder on the bot.

### 2.3.3. Buzzer

The buzzer is to provide user interface feedback and ensure that the system is responding to our command. The feedback is vital as it can improve the experience and drives the product stickiness. As the weighing sensor detects the presence of the soda can in the container, the buzzer will buzz and remind us that the payload is ready for delivery. When the TA removes the soda can from the container, the buzzer will buzz again to remind TA that the delivery is done.

## 2.4. <u>Designs</u>

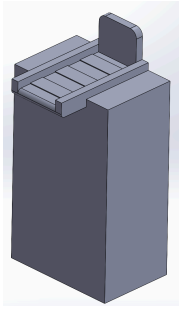
### 2.4.1. Dispenser



**Linear-actuated Piston Dispenser**

The soda can will be placed in front of the piston, in between the rails. When the user has selected the table number, and the turtlebot is in position to receive the can, the piston will be activated and will push the can along the rails until it reaches the cupholder on the bot.

**Conveyor belt Dispenser**



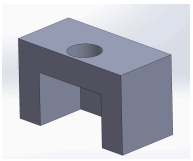
Similar to the piston actuated dispenser, this design will instead use a conveyor belt to facilitate the movement and transfer of the can into the turtlebot. The drinks can can be placed anywhere along the belt and once the bot is in place, the motor of the conveyor will activate until the can is off the belt. However, this technology can be both complex and expensive. Moreover, the conveyor belt cannot extend past the dispenser, so the can cannot be smoothly transferred to the turtlebot.

**Trapdoor-based Dispenser**

In this design, the dispenser will act as a dock or garage for the turtlebot where it can go under and inside the dispenser. The can will be loaded into the hole. When the user has selected the table number, and the turtlebot is in position to receive the can, a

trapdoor will open and allow the can to fall by gravity onto the bot. Although this method would be relatively simple to build and implement, there is a risk of it failing due to structural or material failure every time the can is dropped from height, leading to uncertainty in the feasibility and longevity of this concept.





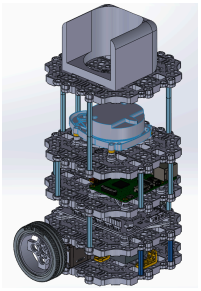
### 2.4.2. Modifications to Turtlebot to receive the payload



We took into account quite a lot of possible ways to efficiently settle in the can after it is dispensed into the delivery bot. Following are some of them:

**Fifth Waffle Concept**

In this concept, we are building upon the design features of the turtlebot3 burger by adding another layer to it and using the extra layer to fit our needs. A custom designed “cup holder” will be secured on this fifth waffle for the soda can to sit in as well as to accommodate the loading of the can from the dispenser. This design will also be more stable because the consequences of the higher centre of gravity is not as great as the can will be securely placed inside the cupholder.



**Piggyback Concept**

This concept involves a cupholder being attached to the back of the turtlebot. This ensures that the can is well-secured and does not fall off the bot during motion. Moreover, the Lidar will be unobstructed. However, the con of this design is the uneven distribution of weight once the can is loaded, the bot may become unstable and trip over.

## 3. Concept of Operations

### 3.1 Overview of Phases

We have divided the mission into three phases, phase 1 deals with the dispensing of the soda can into the turtlebot along with the input of the designated table number using the button system on the dispenser. The second phase includes detection of load in the turtlebot and the navigation towards the final destination. The third phase includes receival of the order by the customer and the return journey back to the dispenser where it will be ready for the next order.

### 3.2 Phase 1 - Dispensing Phase

1. **Objective of this phase:** To successfully dispense the can from the dispenser to the turtlebot for delivery.

2. **Description of the phase:** Once the dispenser has been loaded with a soda can manually, the can will be pushed into the turtlebot such that it does not fall out. This phase involves receiving the table number via the number pad on the dispenser.
   Once the table number is sent from the dispenser's ESP32 to the bot's Raspberry Pi via Bluetooth and the NFC module detects the NFC tag on the bot, the linear actuator is activated and the can is pushed onto the bot. When the HX711 weight sensor detects an increase in value, the bot will start navigation towards the pre-set table.

### 3.3 Phase 2 - Search/Navigation Phase

1. **Objective of this phase:** To navigate to the required table while avoiding obstacles.
2. **Description of the phase:** We will use Dijkstra's algorithms to calculate the shortest path, assuming there are no unknown obstacles.
   The LIDAR-SLAM system will dynamically generate the latest map and the new map will be compared to the pre-loaded map for discrepancies. If there is a discrepancy because of a newly introduced obstacle, we will recalculate the path using the same Dijkstra algorithm. If there are no new obstacles, it will continue on its initial path. This process will continue until the bot reaches its destination

### 3.4 Phase 3 - Delivery and the return journey

1. **Objective of the phase:** This phase includes the return journey of the turtlebot back to the dispenser for the next order.
2. **Description of the phase:** Autonomous navigation back to the dispenser will follow the same process as described in phase 2. When the bot reaches the dispenser, the NFC tag on the bot will be detected by the PN532 NFC module. On detection, the bot will await for table information to be sent from the dispenser, signalling a new order. When both table information is keyed in and NFC is detected, the bot is ready to accept a new payload. The linear actuator is triggered and the new can is pushed onto the bot, starting a new order. **(Appendix C)**

## 4. Preliminary Design

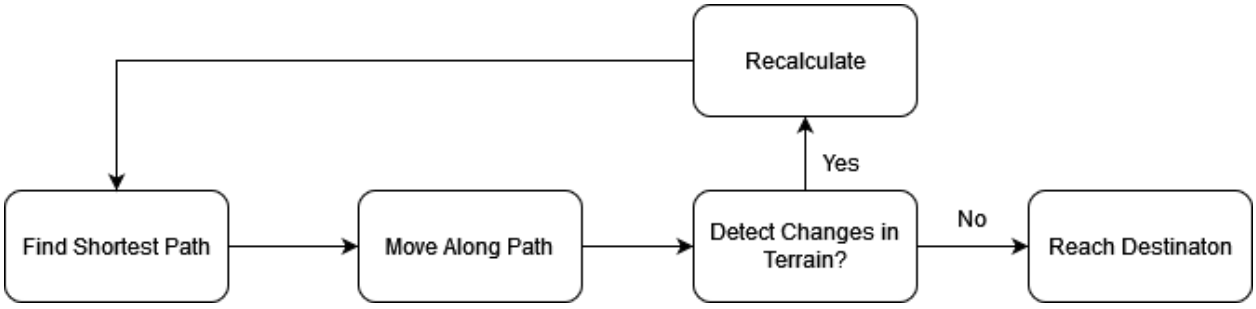
After careful consideration of the various systems and designs in our literature review, we have chosen the following:

### 4.1. Overview of Systems

| No. | Purpose | Description | Method |
|---|---|---|---|
| 1 | Autonomous Navigation of the turtlebot | - Determines the location of the robot<br>- Able to navigate itself to the designated table number | - Dijkstra Algorithm for finding the shortest path.<br>- Obstacle avoidance algorithm<br>- LIDAR |
| 2 | Digital Interface between the turtlebot and dispenser | - Provide the table number to the bot for navigation | - Numpad<br>- ESP32 microcontroller<br>- PN532 NFC |
| 3 | Physical Interface of the entire system | - Deliver the payload from the dispenser to the robot | - Load cell and HX711<br>- Buzzer |
| 4 | Designs | - Making feasible yet efficient components | - Piston Actuated Setup<br>- Fifth Waffle Design |

**Autonomous Navigation**
We will be employing Dijkstra's algorithm for navigation and obstacle avoidance. The ease of implementation of this algorithm compared to A* and D* greatly contributed to our decision. In light of limited time and knowledge, we decided that the easiest algorithm to implement is the way to go. Although its time complexity is the greatest, the room that our mission will be conducted in is quite small and thus the obstacles that the bot will face is also quite limited, resulting in only a small difference in time efficiency. Moreover Dijkstra Algorithm is apt as we can treat the search as an uninformed search when taking into account unknown obstacles. If Dijkstra proves to be too slow, we will consider implementing the A* algorithm.

**Obstacle Avoidance**
Recalculation of the shortest path will happen whenever there is a discrepancy between the pre-loaded map and the dynamically loaded map that is actively scanned by the LIDAR mounted on the bot. Consistent mapping and polling is necessary to successfully avoid obstacles. If the mapping appears to slowly, we will consider slowing the bot down.

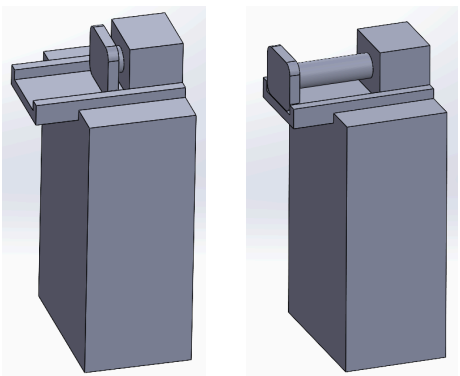



**Microcontroller (dispenser)**
ESP32 will be our choice for the dispenser's microcontroller. All 3 boards have very similar functions, possessing wireless capabilities, GPIO pins and low power requirements. Due to limited budget, the ESP32 is the most favoured board. A quick search on Amazon.sg will highlight the price difference. **(Appendix D)**

**PN532 RFID NFC Module**
We will be using PN532 over the RC522 module as it supports not only NFC on top of RFID, it can also handle more communication protocols. The added flexibility and capabilities at a similar price point drew us to this module. The NFC will need the bot to park very precisely beside the dispenser. When that happens, we will make use of RFID will navigate the bot towards the dispenser as RFID can be used to measure distance.

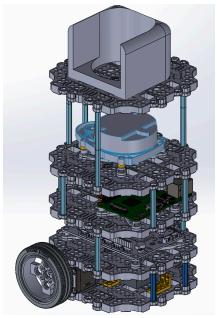
**Communication Protocol between bot and dispenser**
We will be making use of the in-built bluetooth module in both the ESP32 and Raspberry pi. MQTT was a great option but we will need to introduce an extra message broker into the system, which may increase the complexity of our system unnecessarily, considering that we already have bluetooth capabilities. The low band-width of bluetooth might cause the interaction between the controllers to be slow, when that happens, we might consider switching to MQTT

**Dispenser Design: Linear-Actuated Piston Dispenser**
When the order has been placed, the linear actuator piston will start pushing the payload in the container, and it will be set to push the payload in an exact distance for each time to prevent the payload being overly pushed.



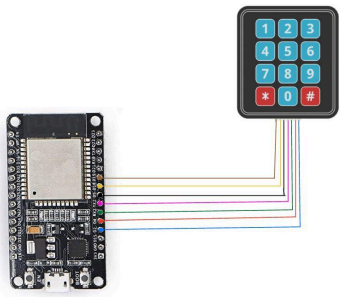
The linear actuator's movement is controlled by a motor driver, which sends two PWM signals to the actuator - one for expansion and one for retraction. The ESP32 uses its PWM pins and the analog write function to set the speed of the actuator. When a button on the Number pad is pressed, the ESP32 sends 5V from Pin D27 to the LPWM on the motor driver, causing the actuator to extend at full speed. Once the maximum distance is reached, the ESP32 sends 5V from Pin D26 to the RPWM on the motor driver, causing the actuator to retract at full speed. **(refer to Appendix E)**

**Turtlebot Design: The Fifth Waffle Concept**



The can will be placed into the cupholder that is fixed on top of a fifth waffle which in turn is secured on top of the fourth waffle of the turtlebot burger. The circumference of the groove is cut out such that the can sits perfectly inside it and hence will be stable.
The support pillars might obstruct the LIDAR, however, we can programme it to ignore the specific angles at which the pillars of the fifth waffle are inserted.

**Number Pad:**

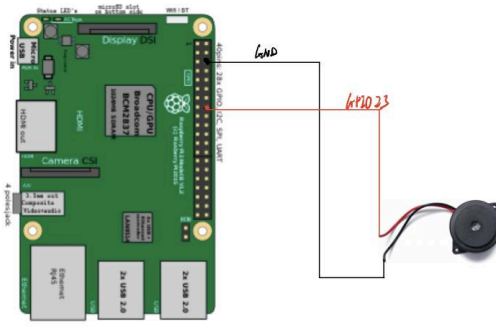



When the table's number is pressed on the number pad. A signal transmitted to the ESP32 controller. The ESP32 can then send the table information to the Raspberry Pi via bluetooth.
There will be 6 pins on the NumberPad and they can be categorised to 2 groups: rows and columns. All the pins will be connected to the GPIO pins on the ESP32 and by defining the GPIO pins in the coding system, NumberPad can provide the table number to the ESP32. **(refer to Appendix F)**

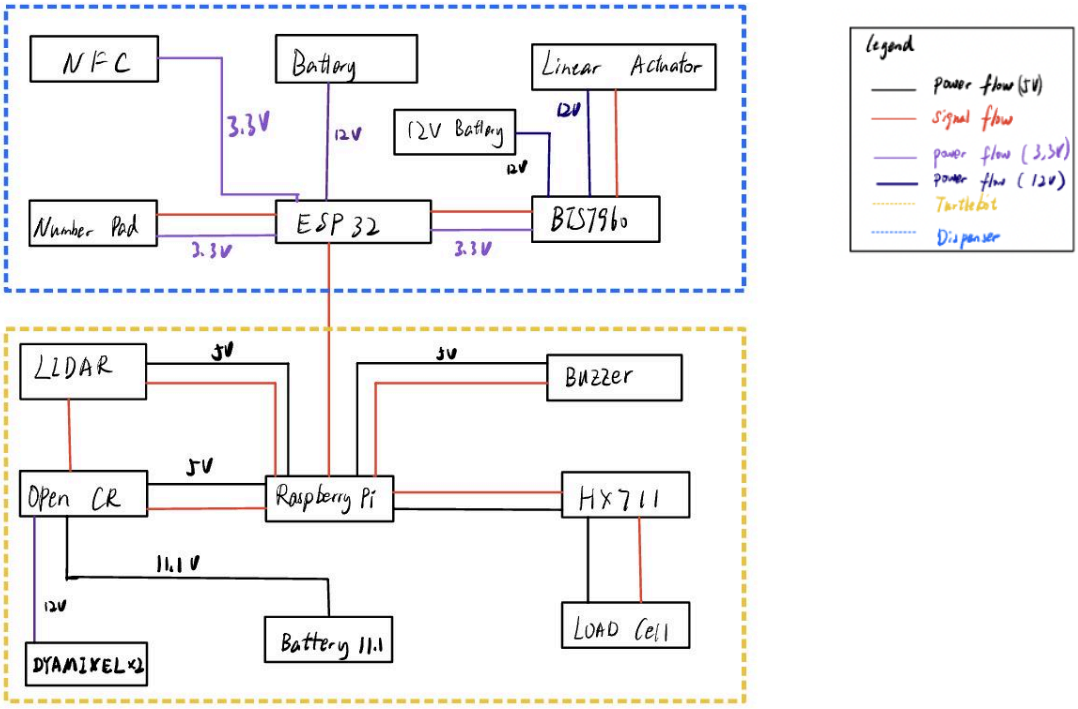
**Weight Sensor:**
To let the robot know that the soda can has been dispensed into the cupholder successfully, a load cell with the amplifier module HX711 will be placed at the bottom of the cupholder. When the can is in the cupholder, the load cell senses the pressure of the can and produces an electrical analog voltage to the HX711, which will amplify and convert the load cell output into digital form for the Raspberry Pi to process, signalling that the payload is received.

- Additionally, the reference weight in the weighing sensor should be adjusted to a number that is marginally less than the average weight of the payload so that the robot can detect when the payload is prepared for delivery by placing it in the container.
- Prior to the robot delivery, a delay time setting should be made to ensure that the payload has settled. **(refer to Appendix G)**

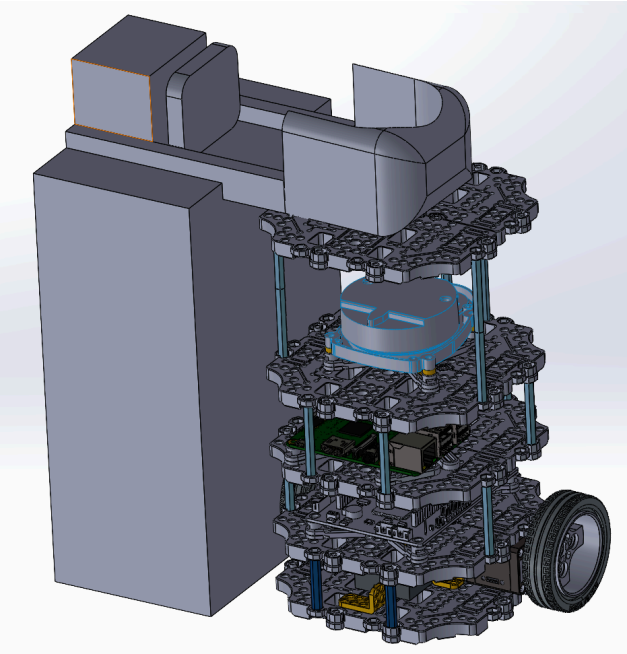
**Buzzer Connection:** To connect the buzzer to the weighing sensor, connect the GPIO pin 23 to the buzz's GND. Once the weight sensor detects that the container has a payload, it will signal the Raspberry Pi to activate the buzzer and produce a buzzing sound for half a second. When the sensor detects that the payload has been removed, the buzzer will sound again.



## 4.2. Functional block diagram

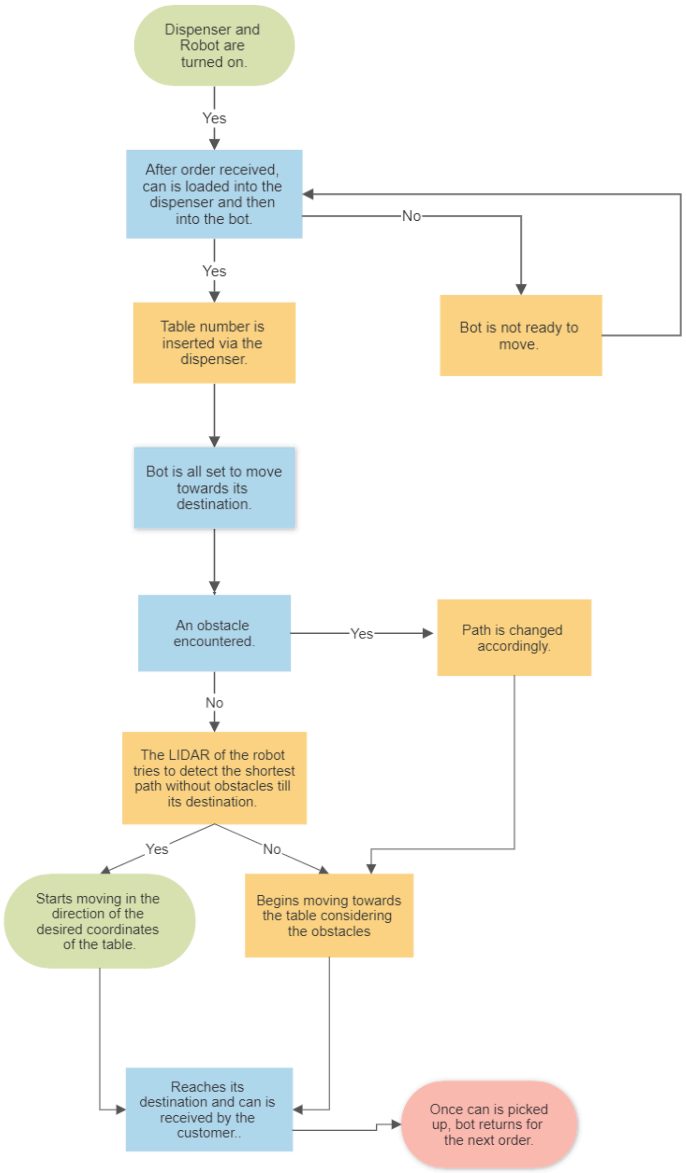



## 4.3. CAD Mockup

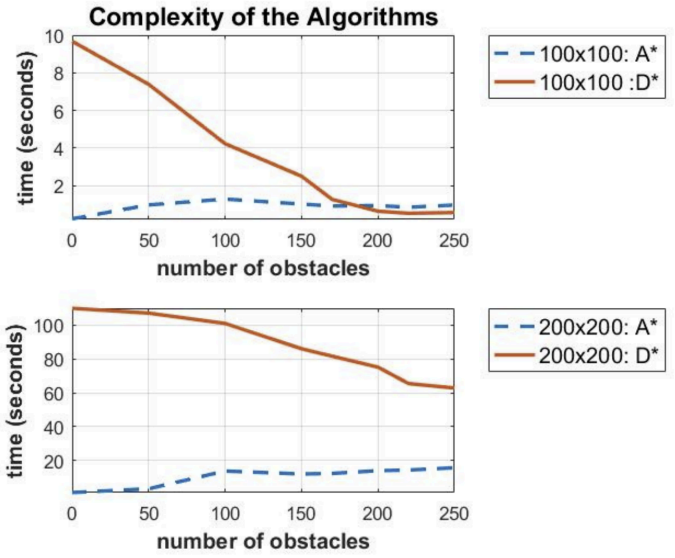


# Appendix A

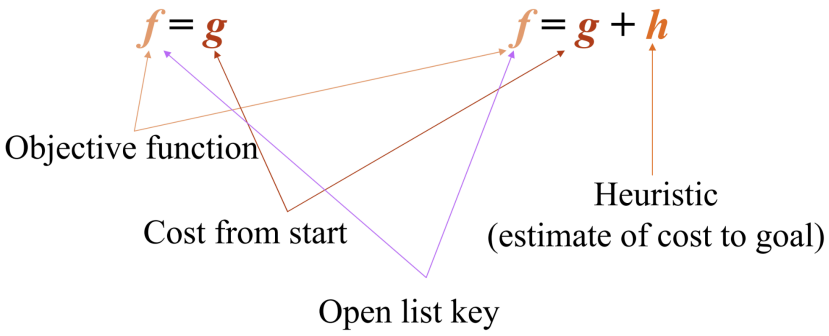
## Mission flowchart



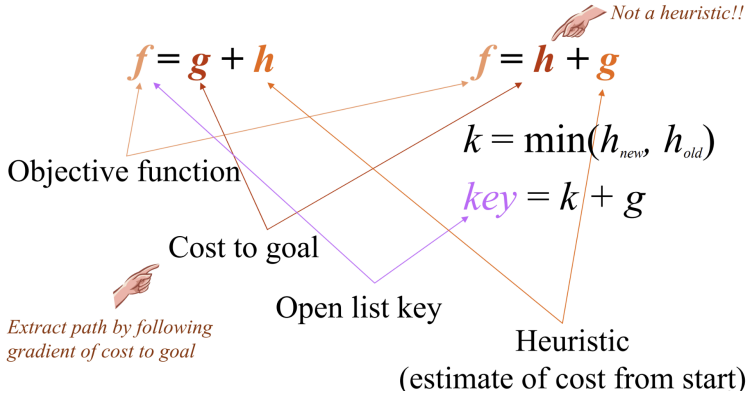
## Appendix B



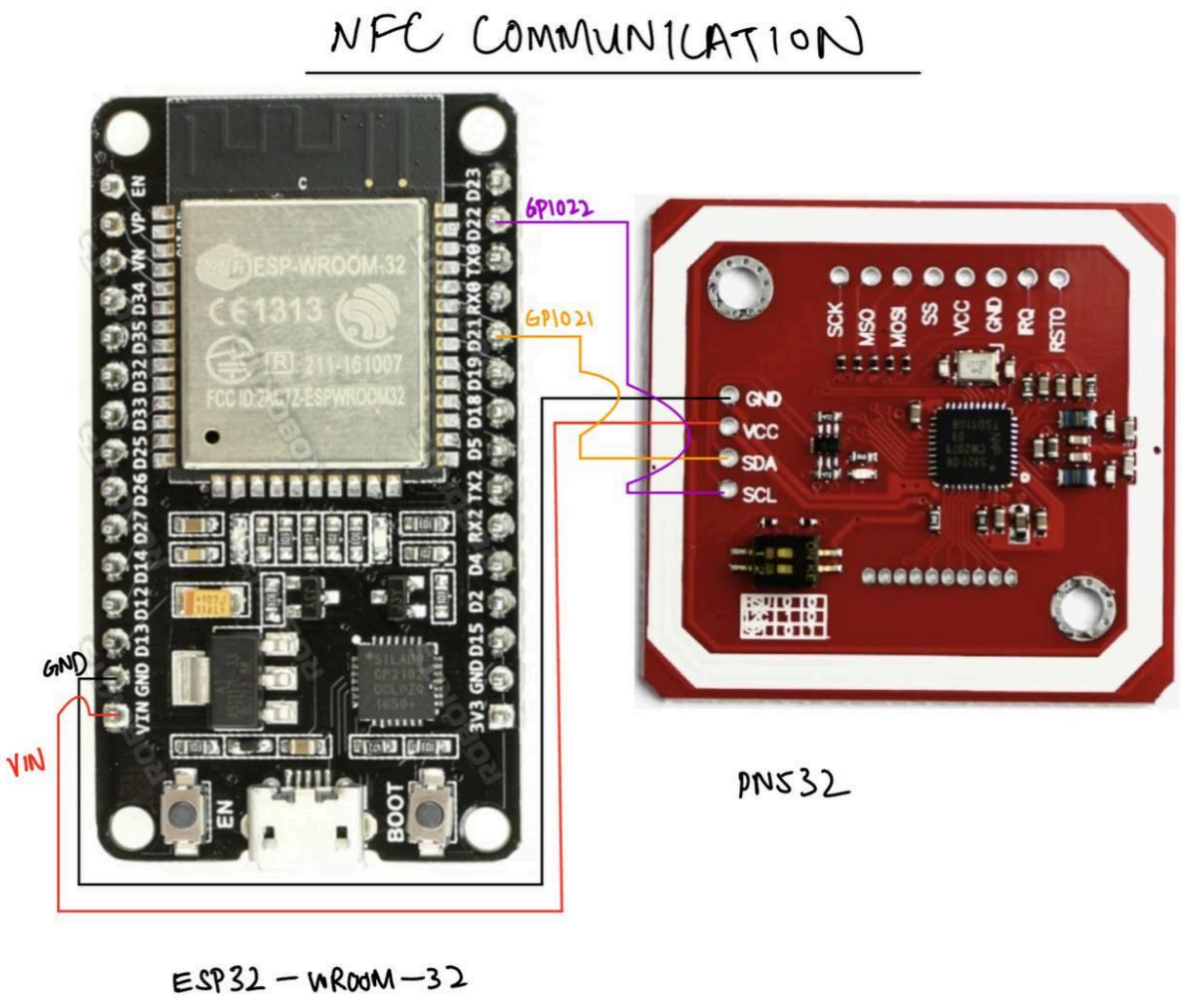

**Dijkstra's Algorithm** **A***



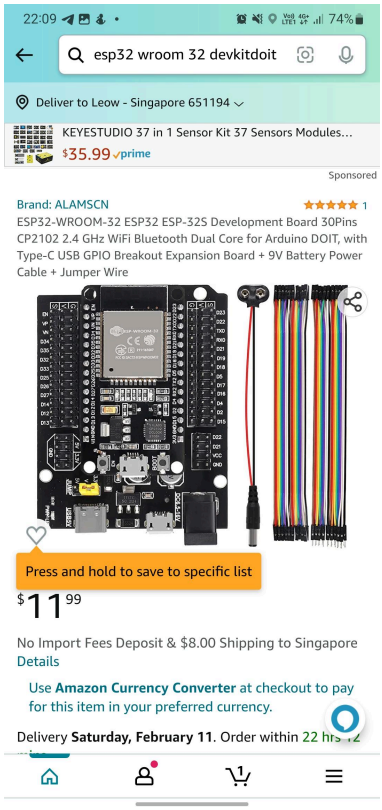

**A*** **Focused D***



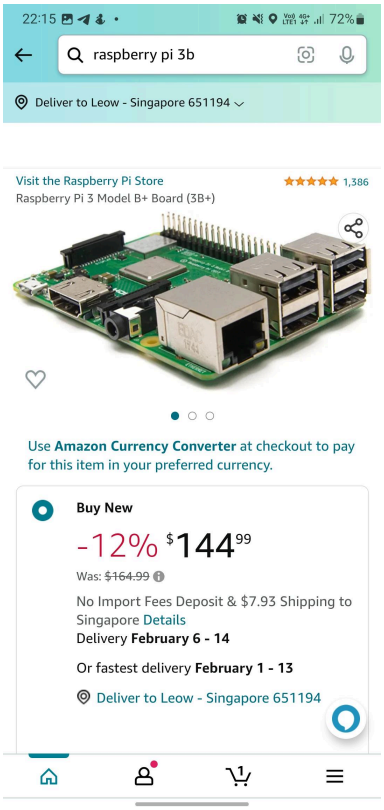
**Appendix C**


NFC COMMUNICATION
ESP-WROOM-32
GP1022
GP1021
GND
VIN
EN
BOOT
SCK
MSO
MOSI
SS
VCC
GND
RQ
RSTO
GND
VCC
SDA
SCL
PN532
ESP32 – WROOM – 32

# Appendix D

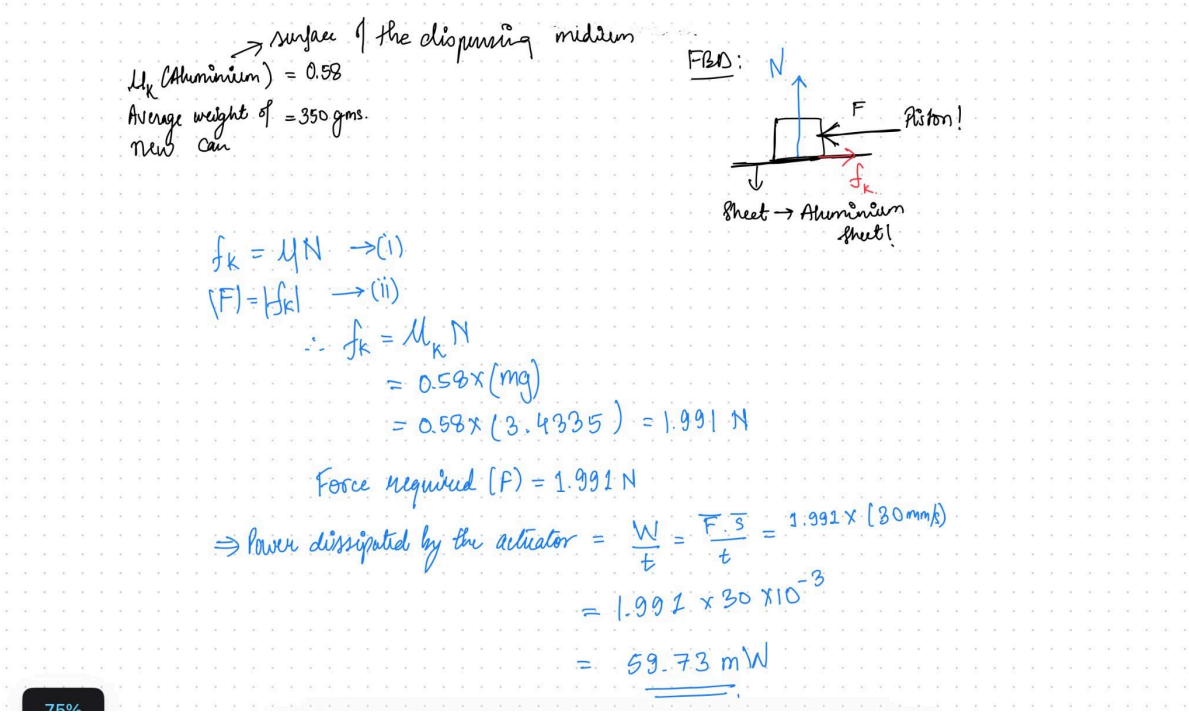

22:09
74%
esp32 wroom 32 devkitdoit
Deliver to Leow - Singapore 651194
KEYESTUDIO 37 in 1 Sensor Kit 37 Sensors Modules...
$35.99 prime
Sponsored
Brand: ALAMSCN
1
ESP32-WROOM-32 ESP32 ESP-32S Development Board 30Pins CP2102 2.4 GHz WiFi Bluetooth Dual Core for Arduino DOIT, with Type-C USB GPIO Breakout Expansion Board + 9V Battery Power Cable + Jumper Wire
Press and hold to save to specific list
$11 99
No Import Fees Deposit & $8.00 Shipping to Singapore
Details
Use Amazon Currency Converter at checkout to pay for this item in your preferred currency.
Delivery Saturday, February 11. Order within 22 hrs 12

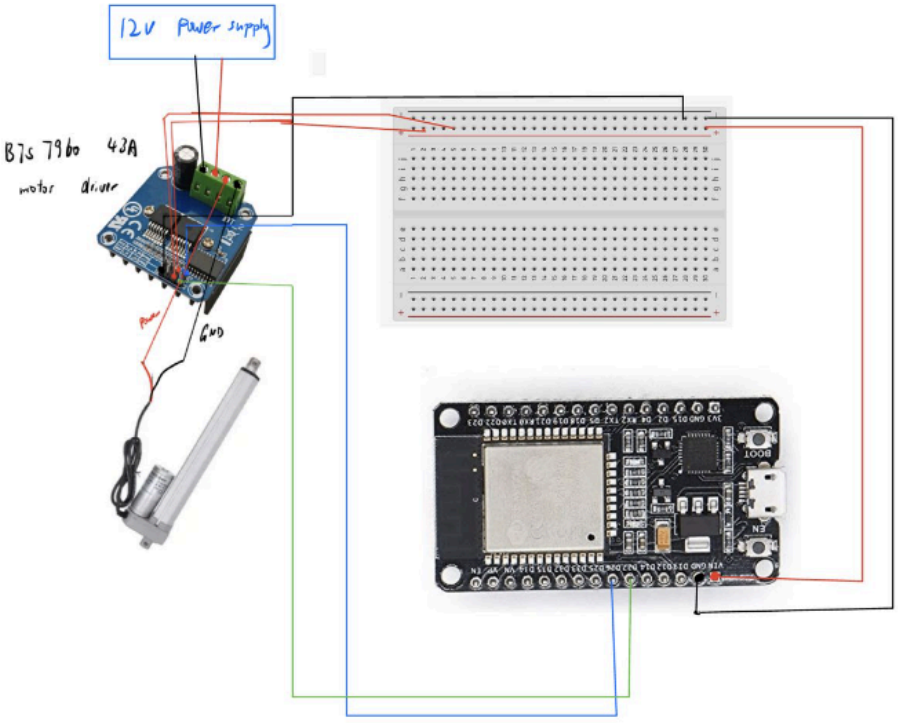

22:12
73%
arduino uno
Deliver to Leow - Singapore 651194
Arduino Mega 2560 REV3 [A000067]
$48.27 prime
Sponsored
Brand: Arduino
7,795
Arduino UNO REV3 [A000066]
#1 Best Seller in Robotics
Size: Arduino Uno regular
-5% $28 50
List Price: $29.95
No Import Fees Deposit & $7.85 Shipping to Singa
Details


22:15
72%
raspberry pi 3b
Deliver to Leow - Singapore 651194
Visit the Raspberry Pi Store
1,386
Raspberry Pi 3 Model B+ Board (3B+)
Use Amazon Currency Converter at checkout to pay for this item in your preferred currency.
Buy New
-12% $144 99
Was: $164.99
No Import Fees Deposit & $7.93 Shipping to Singapore Details
Delivery February 6 - 14
Or fastest delivery February 1 - 13
Deliver to Leow - Singapore 651194

## Appendix E

The calculations for the power required for the linear actuator:






```
arduino PWM.ino   new arduino.ino   numpad.ino
```

```ino
byte mspeed=0;
int RPWM = 26; //GPIO 26
int LPWM = 27; // GPIO 27

void setup() {
  // put your setup code here, to run once:
   Serial.begin(9600);

   pinMode(RPWM,OUTPUT);
   pinMode(LPWM,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if( key ){
    delay(2000)
    mspeed = 255;
    analogWrite(RPWM,0);
    analogWrite(LPWM, mspeed);
    delay(1000)
    mspeed = 255;
    analogWrite(RPWM,mspeed);
    analogWrite(LPWM, 0);
  }
  else{
     analogWrite(RPWM,0);
    analogWrite(LPWM, 0);
  }

}
```

## Appendix F

amazon Deliver to Singapore | All | Search Amazon | EN | Hello, Acco

All | Today's Deals | Customer Service | Registry | Gift Cards | Sell

KEYESTUDIO 37 in 1 Sensor Kit 37 Sensors Modules Starter Kit for Arduino Mega R3 2560 Raspberry Pi Programming, Electronics Components STEM Education Set for Kids Teens Adults + Tutorial

99

Electronics › Computers & Accessories › Computer Components › Single Board Computers


Roll over image to zoom in

### DIYables 3x4 4x4 Membrane Matrix Keypad for Arduino, ESP32, ESP8266, Raspberry Pi

Visit the DIYables Store

★★★★★ 1 rating

$9.45

No Import Fees Deposit & $7.92 Shipping to Singapore Details

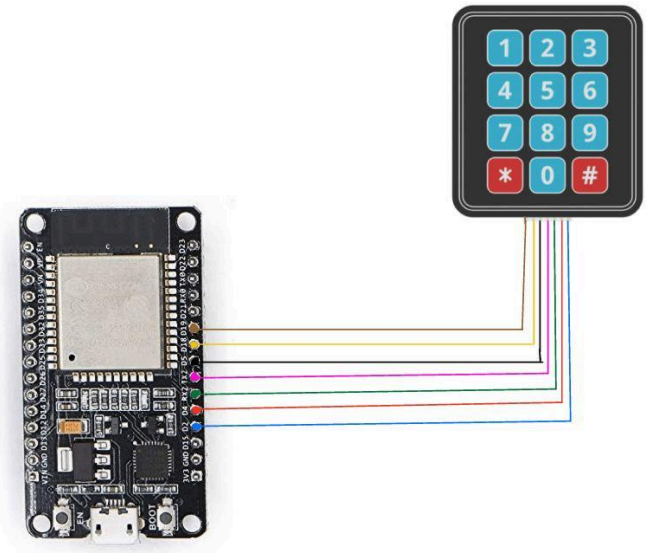
| | |
|---|---|
| **Brand** | DIYables |
| **Keyboard Description** | Membrane |
| **Number of Keys** | 12 |
| **Number of Buttons** | 16 |

**About this item**

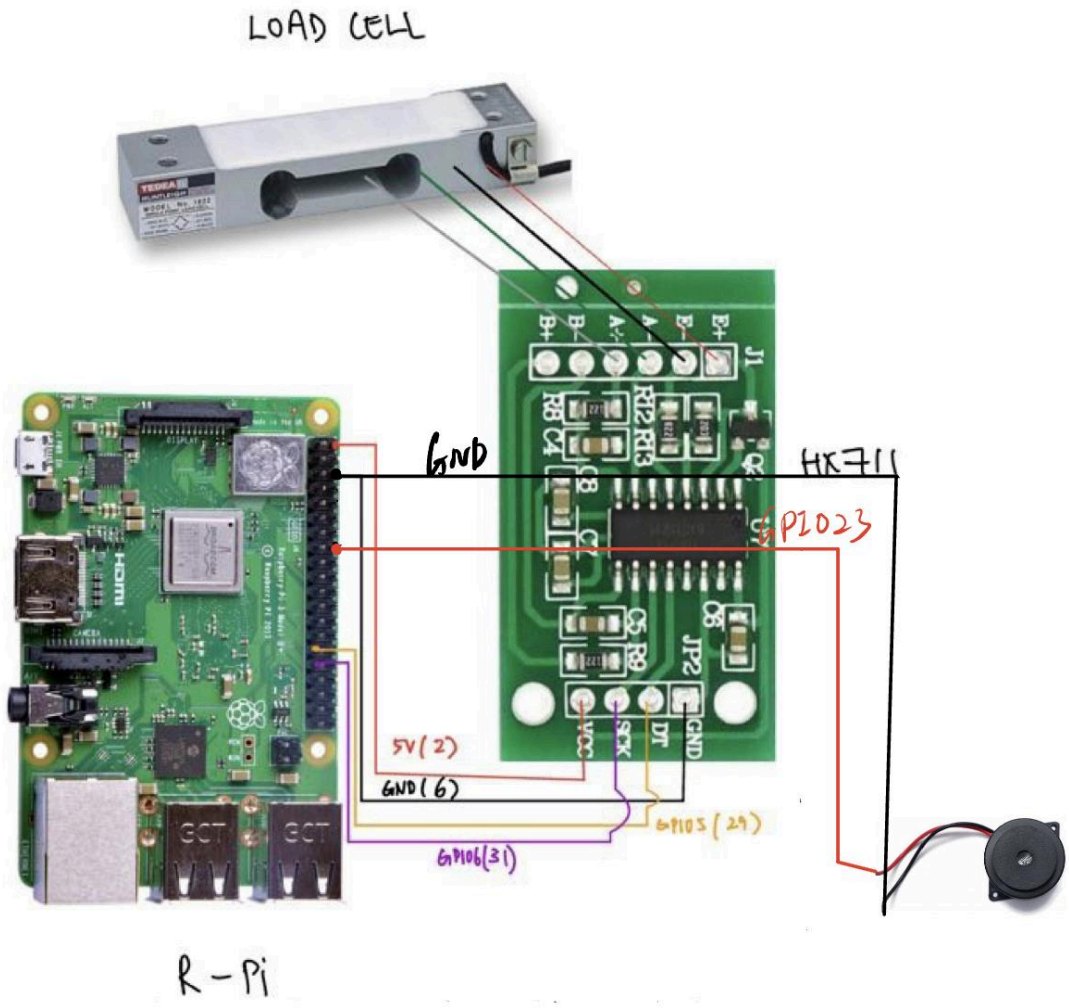
- 3x4 keypad (12 keys) and 4x4 keypad (16 keys)
- Membrane matrix keypad
- Removable adhesive paper on the back => easy to stick on the flat surfaces
- Keypad for Arduino, ESP32, ESP8266, Raspberry Pi
- Tutorials for Arduino and ESP32 are provided

**Note**: Products with electrical plugs are designed for use in the US. Outlets and voltage differ internationally and this product may require an adapter or converter for use in your destination. Please check compatibility before purchasing.

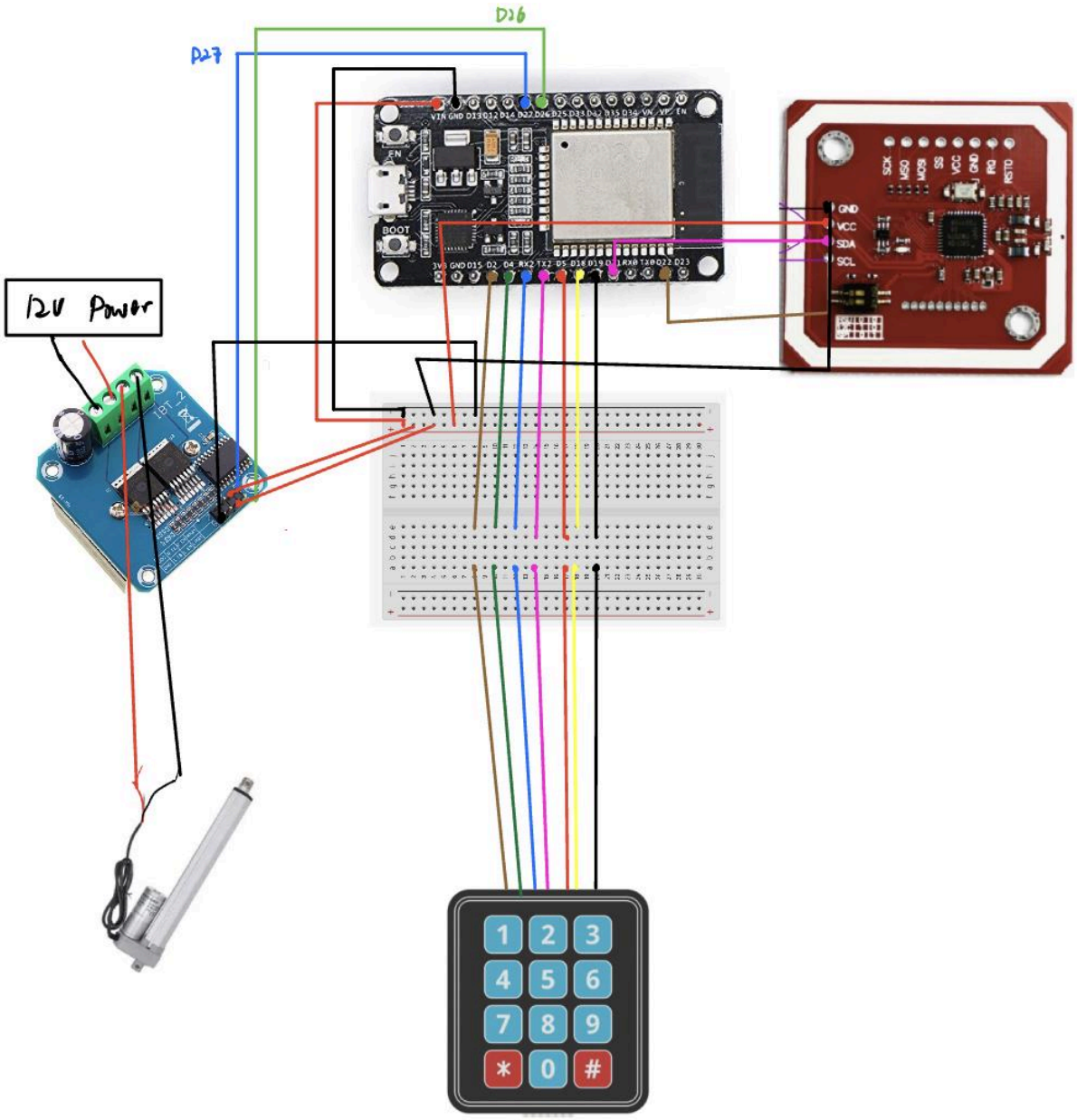
ee2111A lec2 1.16.pdf



**Appendix G**




Circuit diagram for the ESP32


D16
D27
12V Power
BOOT
EN
GND
VCC
SDA
SCL
1 2 3
4 5 6
7 8 9
* 0 #