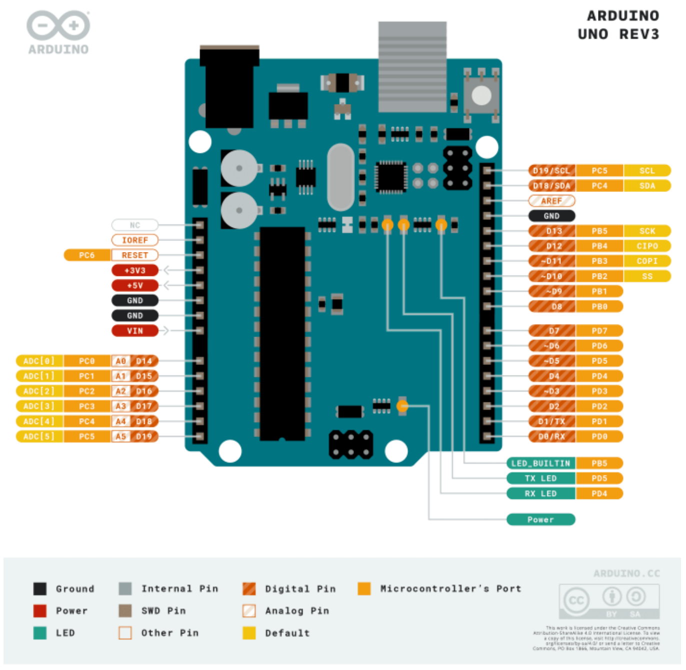
Designing a Line Following Robot to Deliver Goods in a Warehouse using Embedded Systems

There are many microcontrollers that can be used to build a line following robot, and the best choice will depend on specific requirements and constraints, Due to its popularity and wide use as a microcontroller board for DIY and educational projects, the Arduino Uno is a fantastic choice for a line-following robot. It has several qualities that make it an excellent option for a line-following robot.



[1]

- Cost: The Arduino Uno is reasonably priced, making it a viable option for a DIY project with a limited budget.

- Utilization: The Arduino Uno offers an intuitive platform that is simple to use, making it simple to get started even for beginners with little programming experience. Setting up the robot is simple due to the abundance of example and library code available for line following.

- Support: The Arduino Uno has a sizable and vibrant community, so there are numerous tools like tutorials, sample code, and forums where you can get assistance if you encounter any problems.

- Processing Speed: The Uno board's processing speed is sufficient to handle a straightforward line-following algorithm and sensor readings.

In addition, a number of sensors, such as line sensors that can be used for line following, are made to be compatible with the Arduino. This makes it simple for me to locate sensors that your robot can use.

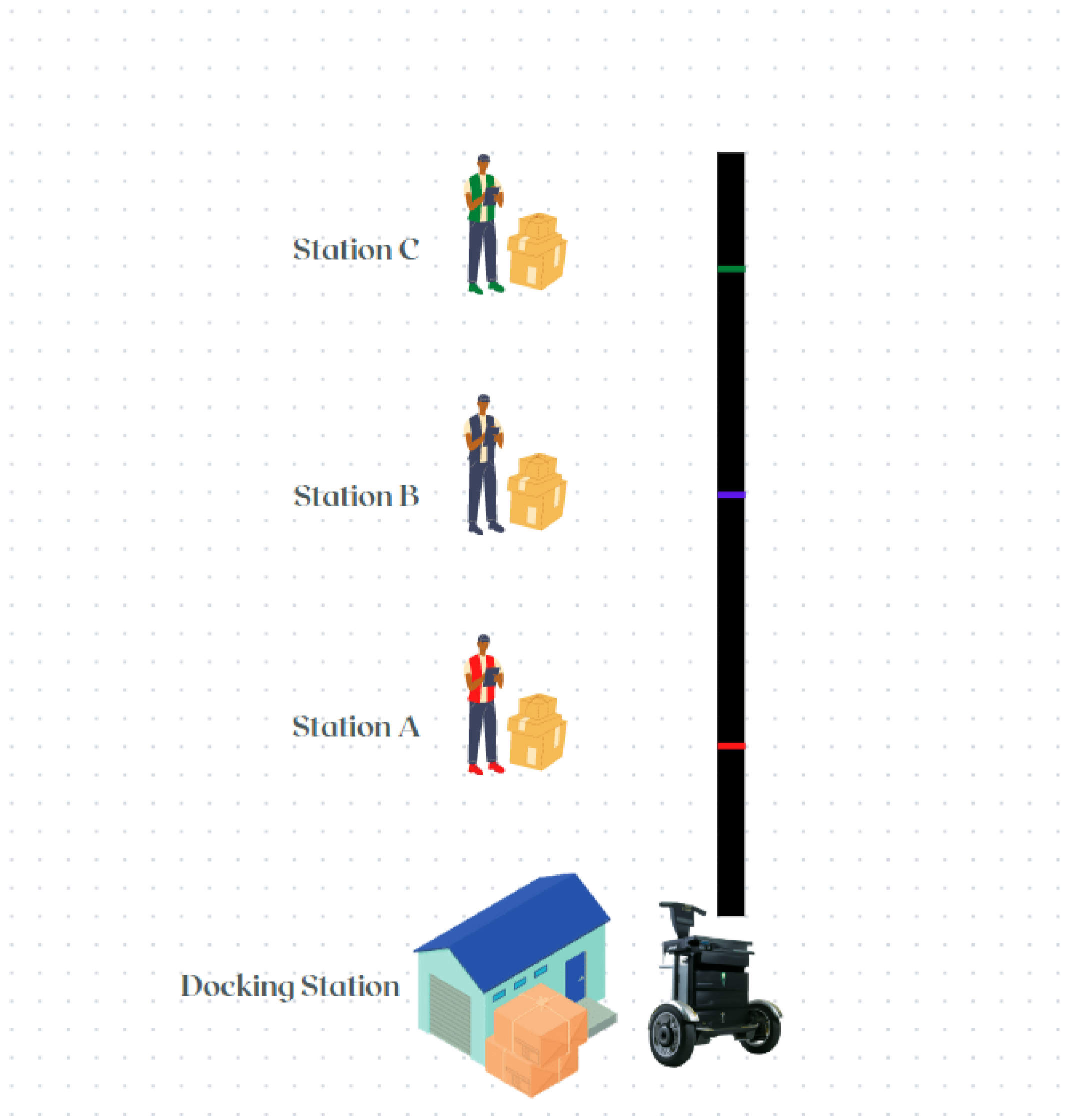
Although there are various microcontrollers that might be utilized for this application, the Arduino Uno might be an excellent fit for my needs.

There are several different types of proximity sensors that can be used with an Arduino Uno, including:

- Infrared (IR) sensors: These sensors emit infrared light and gauge how much of it is reflected. They are frequently employed in the detection of obstacles and line-following robots.
- Ultrasonic sensors: These devices gauge distance using sound waves. They are frequently employed in applications for obstacle identification and ranging.
- LIDAR sensors: Like ultrasonic sensors, lidar sensors detect objects with the use of laser beams. They are typically employed in high accuracy applications like autonomous navigation and 3D mapping.
- Inductive and capacitive proximity sensors: These are contactless sensors that sense changes in capacitance or inductance to determine the presence of an object. They are frequently employed in both home automation and industrial automation.
- Optical sensors: To detect the presence of an item, these sensors typically use a light-emitting diode (LED) and a photodiode. They are frequently employed in programs like item identification and counting.
- Sensors that detect magnetic fields are known as magnetic sensors. They are frequently employed in tasks like locating a rotating shaft or determining the presence of a magnetic card or tag.

Using the right libraries, any of these sensors can be connected to an Arduino Uno, but it's crucial to keep in mind that the Lidar sensors typically demand additional processing power, which an Arduino Uno board might not be able to give, making it more challenging to connect. The Arduino IDE and libraries are readily accessible for the others, and the majority of them may be simply interfaced and programmed using them.

So, this my idea on how the robot will work



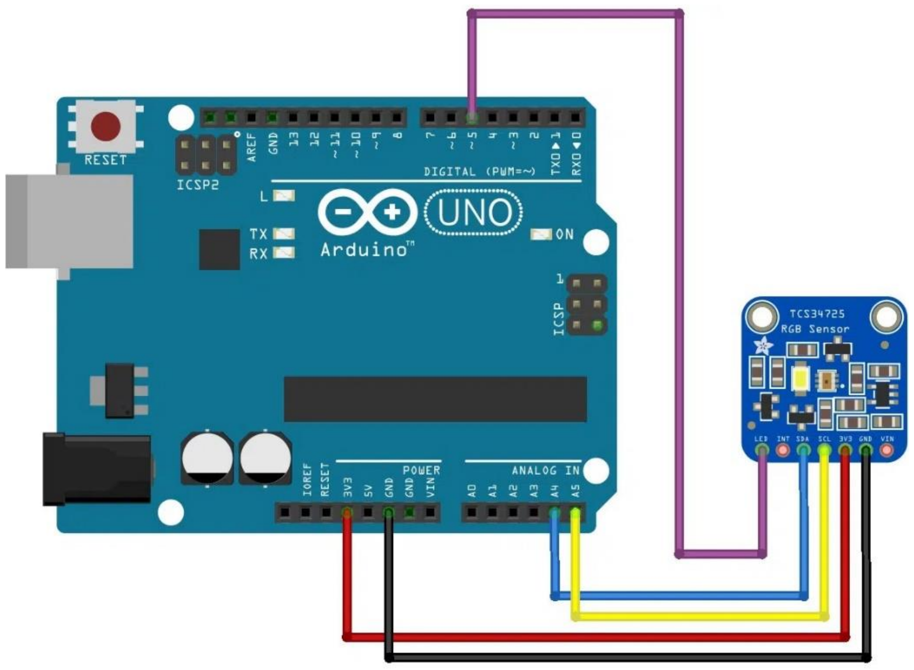
There will be a straight blackline Extending from the docking station all the way up to the station C , on the black line there will be 3 colored stripes to indicate the locations of the stations for station A its Red and for station B its Blue and For station C its Green.

Choosing the right sensors for this project:

1) To Identify colors

So, to identify the location of the stations a color identifying sensor is required there are several color identifying sensors available in the market and I have chosen TCS34725 RGB color sensor, because the TCS34725 is a good choice for an Arduino-based line-following robot because it is a color sensor that can detect a wide range of colors, including red, green, and blue. This allows the robot to accurately identify different colored stripes on the line it is following. Additionally, the TCS34725 is an I2C device, which means it can easily communicate with the Arduino over a two-wire interface, reducing the number of pins required to interface with the sensor. This can simplify the wiring and reduce the overall complexity of the robot.

This is how the wiring should be done,


RESET
ICSP2
AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX▶1 RX◀0
DIGITAL (PWM=~)
UNO
Arduino
L
TX
RX
ON
ICSP
IOREF RESET 3V3 5V GND GND VIN
POWER
ANALOG IN
A0 A1 A2 A3 A4 A5
TCS34725
RGB Sensor
LED INT SDA SCL 3V3 GND VIN

The following code will be used for the Color sensor,

```
#include <Adafruit_TCS34725.h>
// Define constants for the pin connections

const int RGB_SENSOR_SDA_PIN = A4;
const int RGB_SENSOR_SCL_PIN = A5;

// Define color thresholds
const int RED_THRESHOLD = 0xFF0000;
const int BLUE_THRESHOLD = 0x0000FF;
const int GREEN_THRESHOLD = 0x00FF00;


// Define global variables
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);
```

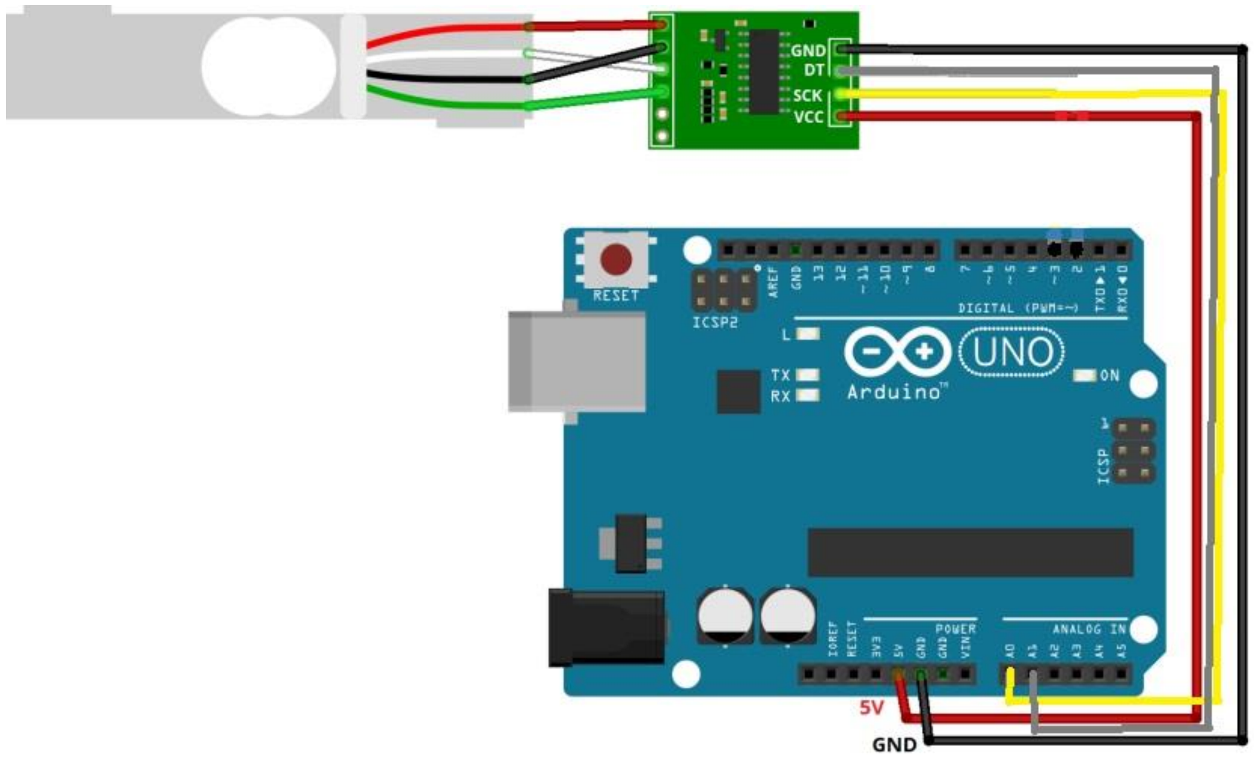
2) To measure the weight

There are several types of weight sensors that are compatible with Arduino, including:

- Load cells: These sensors use a strain gauge to measure weight by converting the pressure or force applied to the sensor into an electrical signal. They are available in a wide range of capacities and are suitable for measuring both static and dynamic weights.
- Pressure sensors: These sensors measure the pressure exerted by an object on the sensor and can be used to determine weight by relating the pressure to the weight of the object.
- Force-sensitive resistors (FSRs): These sensors are similar to pressure sensors but use a resistor that changes resistance depending on the force applied to it.

Load cells are considered as the best option to measure weight within a range of 0-5 Kg because it offers high accuracy, reliability, and repeatability. Load cells are also versatile and can be used in a variety of applications, including industrial weighing, automotive testing, and medical equipment. They are also available in different capacities, sizes, and designs to suit different needs, and easy to interface with Arduino using HX711 load cell amplifiers.

This is how the wiring should be done,



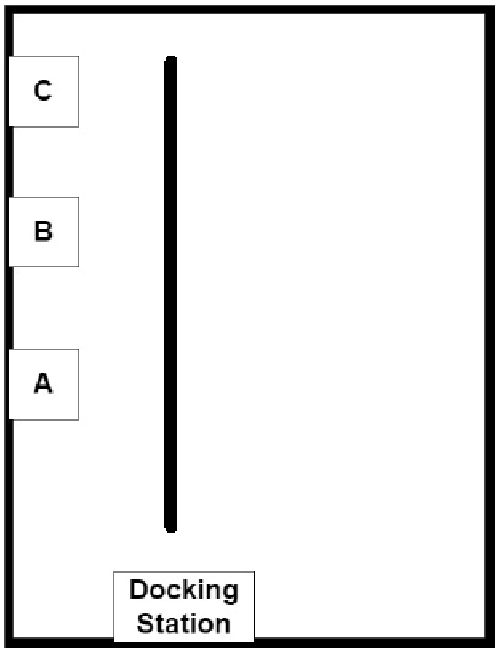
This is the code that is to be written to read the measurements.

```
#include <HX711.h>


const int LOAD_CELL_DT_PIN = A0;
const int LOAD_CELL_SCK_PIN = A1;
// Define global variables
HX711 loadCell;
float weight;
void loop();{
  weight = loadCell.get_units();
```

Based on this code and the readings taken by the loadcell the robot will decide the actions it has to execute .

The color of the line that I chose was black because it will be easier to execute a follow line protocol using 2 IR sensors so that it detects the blackline and follows it.



Warehouse Layout

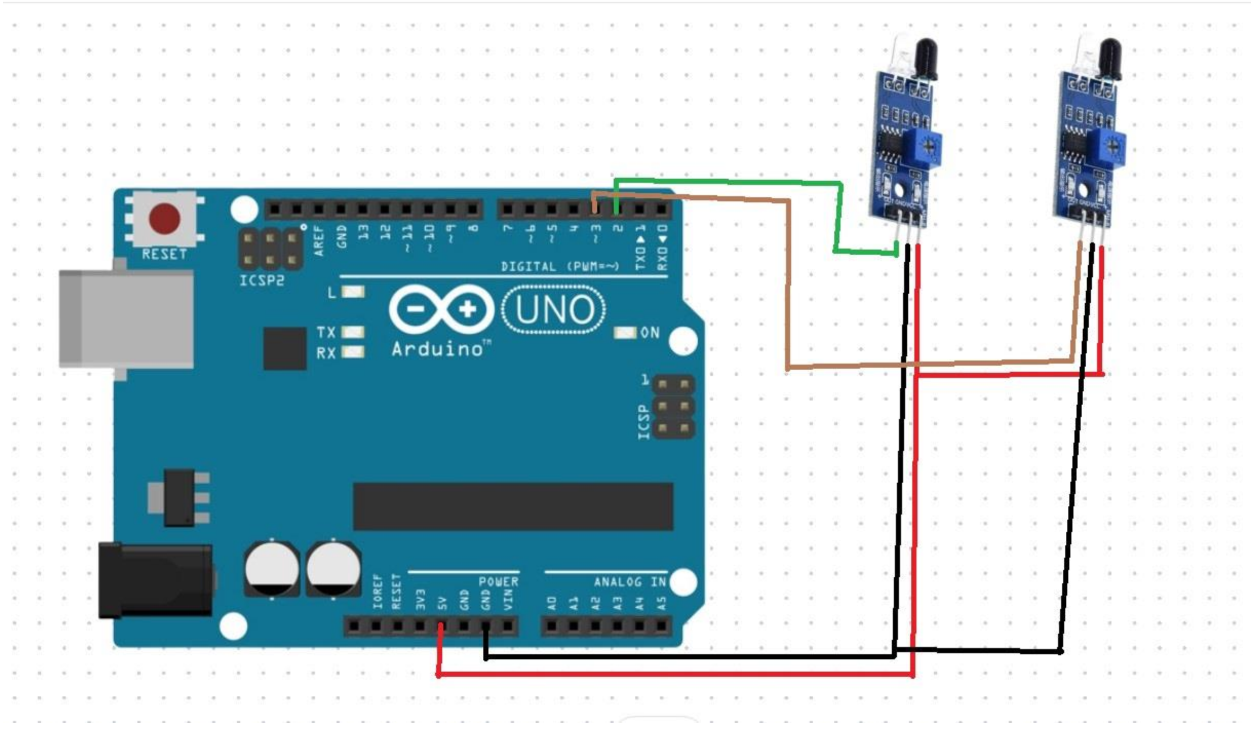
So to detect the line I chose 2 IR sensors, Due to their ability distinguish between a black line and a white surface, IR (infrared) sensors are a popular option for line-following robots. Infrared light from the sensor bounces off the surface and returns to the sensor. The color of the surface can then be determined by the sensor by measuring the brightness of the light that is reflected.

So in this case I choose IR sensor for this project because it has the following advantages over the rest,

1. Infrared (IR) sensors are affordable and simple to use. They don't need any extra calibration or adjustments, and they can be quickly integrated into an Arduino-based line-following robot.
2. Some IR sensors can detect the line even at a distance due to their good sensing range. As a result, the robot is more adaptable and can follow the line in tighter corners and curves.
3. On the other hand, other proximity sensors, such ultrasonic or Lidar sensors, are significantly more complicated and expensive, and they might not be able to detect contrast or lines with the same accuracy and precision. In extreme settings like those found in robot contests or industrial applications, they also tend to be less durable.

In conclusion, IR sensors are a low-cost, simple-to-use, and accurate alternative for line-following robots. They are also simple to integrate into an Arduino-based robot and have good contrast detecting capabilities

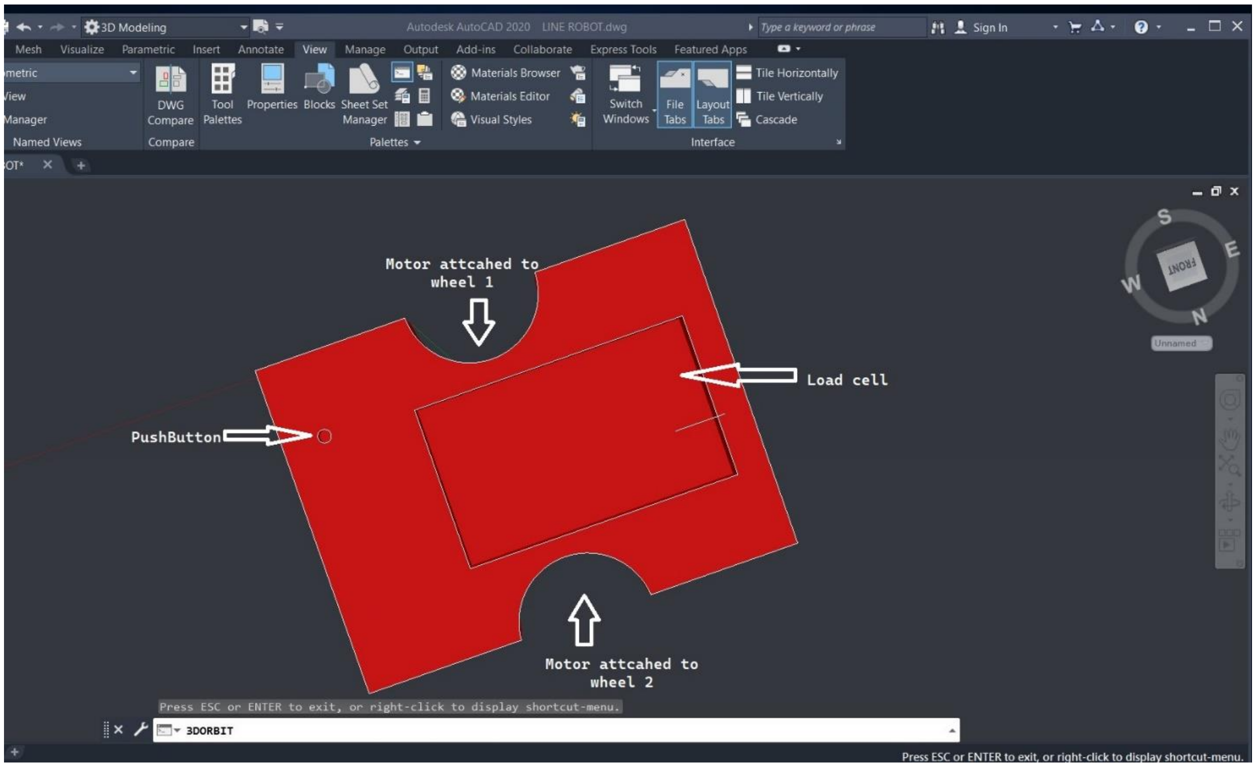
So for this project I choose IR sensors to detect the line and follow the line, This is how the sensors will get integrated with the Arduino Board.



Arduino's 5v pin will be connected with the VCC pins of both IR sensors and the ground pin of Arduino will be connected with both of the ground pins of IR sensors , finally the output pins will be connected with the pins 2,3.

- ❖ Just power the 5V pin with 5V, common ground to GND and then connect the SCL labeled pin (top left) to I2C clock and SDA to I2C data. Those are the only four wires you need to control the entire shield.

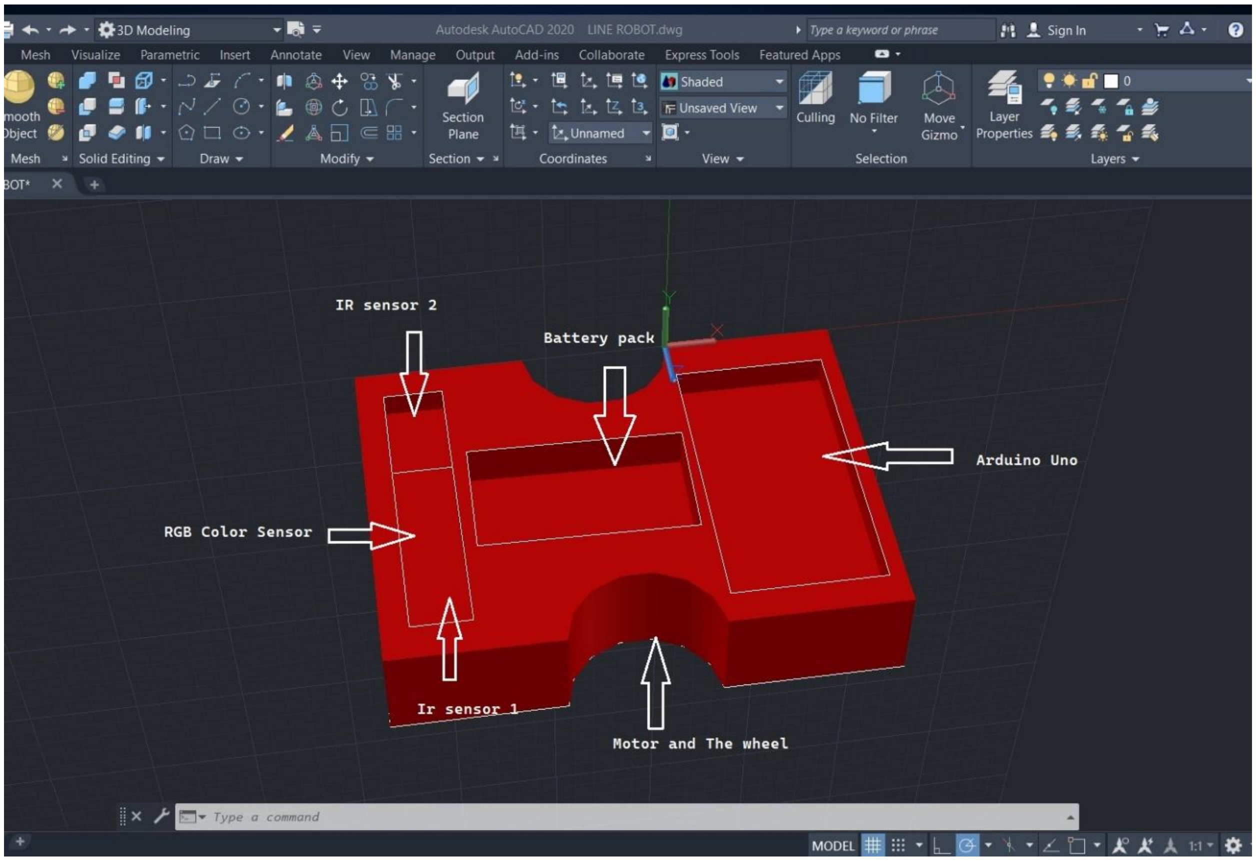
E) Illustrate an appropriate free hand sketch or computer-aided draft of the robot design. Show how the micro-controller board, wheels, and sensors will be installed.


Motor attcahed to wheel 1
Load cell
PushButton
Motor attcahed to wheel 2
Press ESC or ENTER to exit, or right-click to display shortcut-menu.
3DORBIT

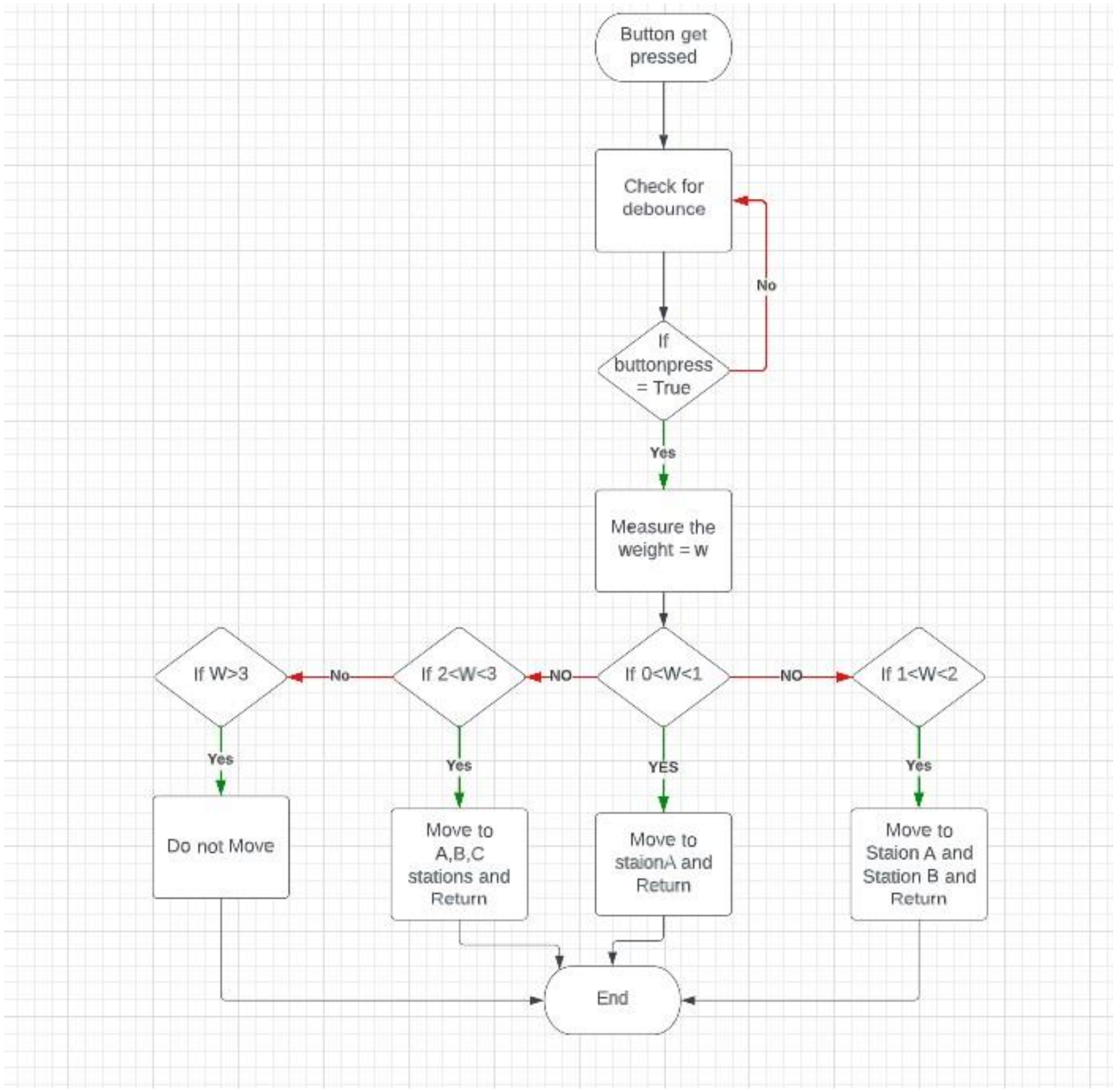
As per my design the motors will be attached to the 2 wheels on the opposite sides and The RFB LCD Shield and the pushbutton will be attached on top and the loadcell will be attached in the area where the loads will be placed.

- ❖ The robot will have a dimension of 20x40 cm

So, in the bottom of the robot the 2 IR sensors will be installed on the corners in front and the color sensor will be placed in the middle and also he main Arduino Uno board and a battery pack will also be installed on the bottom as per the design



The robot will work based on this flow chart



Use the following guidelines to build a fully functional code to operate the robot. Make sure the entire code is readable. You may use arrays, constants, and functions to make the code more organized and comprehensive. Furthermore, it is highly recommended to use comments to deliver an informative code. First, draw a flow chart to depict the algorithm.

1) Write a function to debounce the push button programmatically.

```
#define DEBOUNCE_DELAY 50
// Define constants for the pin connections
const int BUTTON_PIN = 8;
// Define global variables

bool buttonPressed = false;
bool debounce(int pin) {
  bool buttonState = digitalRead(pin);
  if (buttonState != LOW) {
    delay(DEBOUNCE_DELAY);
    if (digitalRead(pin) != LOW) {
      return true;
    }
  }
  return false;
}
```

The constant DEBOUNCE_DELAY is set to 50 milliseconds, and is used as a delay in the debounce() function.

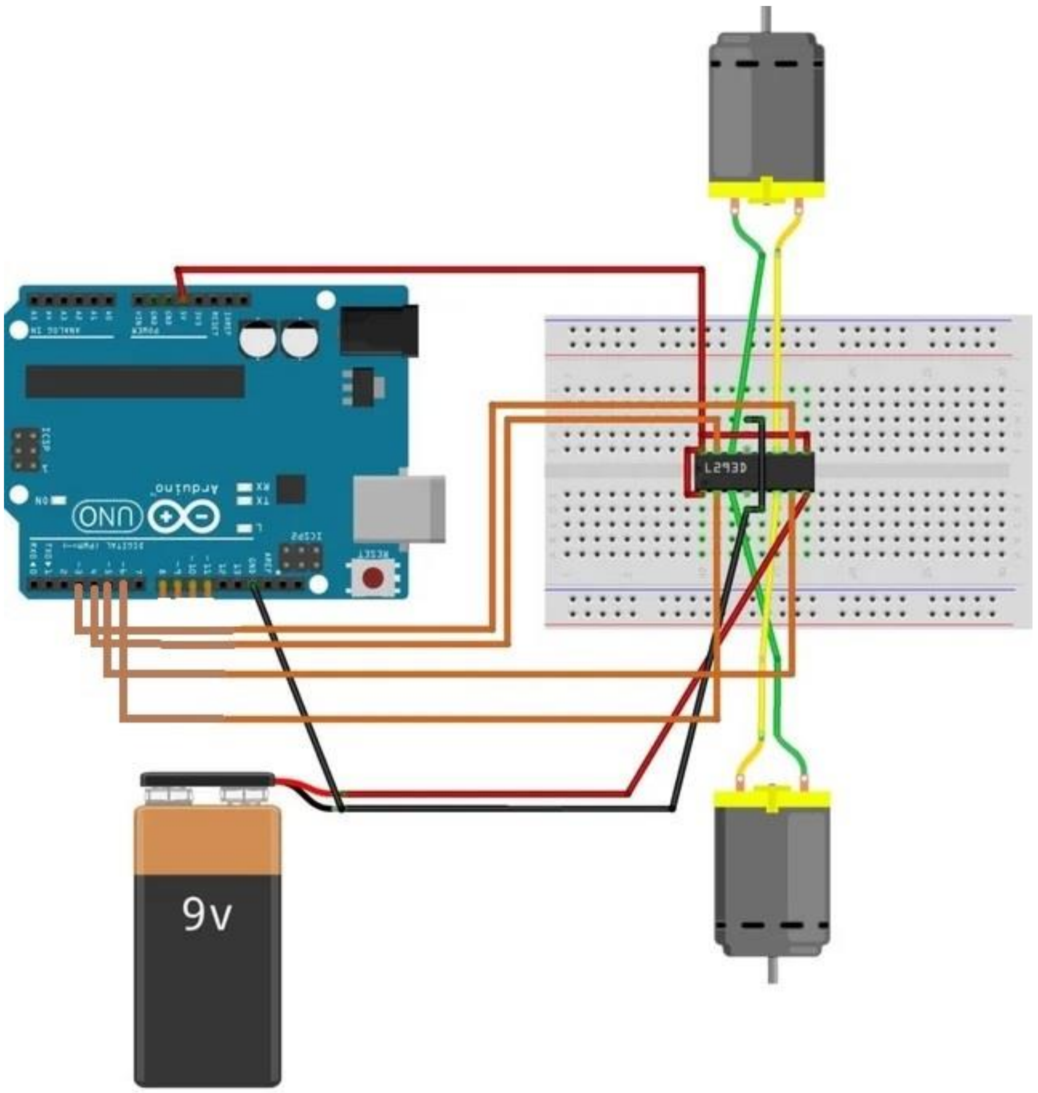
The debounce() function takes an int parameter, pin and checks if the state of the button connected to that pin is not LOW (LOW means not pressed, and HIGH means pressed). If the button is not pressed, the function waits for DEBOUNCE_DELAY milliseconds, and then checks the state of the button again. If the state of the button is still not LOW, the function returns true, indicating that the button has been pressed. If the button state is LOW, the function returns false, indicating that the button has not been pressed.

The global variable "buttonPressed" is defined and initialized as false.

It is a simple implementation of debouncing, which is a technique used to prevent the button from registering multiple presses when it is only pressed once. This is caused by the mechanical nature of the button, which can cause it to bounce between the pressed and not pressed states for a brief period of time after it is first pressed

2) Use online resources to find motor control driver and write proper functions to move forward, turn right/left and stop the robot.

   A motor driver is an electronic circuit or device that controls the speed and direction of a brushless or brushed DC motor. The L293D is a popular motor driver IC that can control the speed and direction of two DC motors simultaneously. The IC has two sets of inputs for controlling the direction of each motor and two sets of outputs for controlling the speed of each motor. The inputs can be controlled using digital signals from a microcontroller or other electronic device. The outputs are connected to the DC motors, which are typically connected to the power supply.



[2]

These are the functions to make it move forward, turn right/left , stop

First we should initialize the L293D Library and define pin connections and set the pinmodes

```
#include <L293D.h>
constants for the pin connections
const int MOTOR_LEFT_PWM_PIN = 6;
const int MOTOR_LEFT_DIR_PIN = 7;
const int MOTOR_RIGHT_PWM_PIN = 5;
const int MOTOR_RIGHT_DIR_PIN = 4;
const int BUTTON_PIN = 8;
 // Set the pin modes for the motors
pinMode(MOTOR_LEFT_PWM_PIN, OUTPUT);
pinMode(MOTOR_LEFT_DIR_PIN, OUTPUT);
pinMode(MOTOR_RIGHT_PWM_PIN, OUTPUT);
pinMode(MOTOR_RIGHT_DIR_PIN, OUTPUT);
```

To move forward, start and set the speed of the motor at a certain level.

```
// Start the motors
  digitalWrite(MOTOR_LEFT_DIR_PIN, LOW);
  analogWrite(MOTOR_LEFT_PWM_PIN, 0);
  digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
  analogWrite(MOTOR_RIGHT_PWM_PIN, 0);
// To move forward
digitalWrite(MOTOR_LEFT_DIR_PIN, LOW);
digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
analogWrite(MOTOR_LEFT_PWM_PIN, 255);
analogWrite(MOTOR_RIGHT_PWM_PIN, 255);
```

To make turns in either Left or Right,

Left Turn,

```
//take left turn
digitalWrite(MOTOR_LEFT_DIR_PIN, HIGH);
digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
analogWrite(MOTOR_LEFT_PWM_PIN, 220);
analogWrite(MOTOR_RIGHT_PWM_PIN, 220);
```

for Right turn,

```
//take right turn
digitalWrite(MOTOR_LEFT_DIR_PIN, LOW);
digitalWrite(MOTOR_RIGHT_DIR_PIN, HIGH);
analogWrite(MOTOR_LEFT_PWM_PIN, 220);
analogWrite(MOTOR_RIGHT_PWM_PIN, 220);
```

And to stop the robot,

```
void stopRobot() {
  digitalWrite(MOTOR_LEFT_DIR_PIN, LOW);
  digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
  analogWrite(MOTOR_LEFT_PWM_PIN, 0);
  analogWrite(MOTOR_RIGHT_PWM_PIN, 0);
  }
```

3) Use the line detecting sensors to follow the line when the push button is pressed. Build an algorithm to follow the line.

```
void followLine(); {
  while (1) {
    //Read the sensor values
    int leftSensor = digitalRead(IR_LEFT_PIN);
    int rightSensor = digitalRead(IR_RIGHT_PIN);

    //Check the sensor values
    if (leftSensor == 0 && rightSensor == 1) {
        //left sensor on black and right sensor on white
         //take left turn
        digitalWrite(MOTOR_LEFT_DIR_PIN, HIGH);
        digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
        analogWrite(MOTOR_LEFT_PWM_PIN, 220);
        analogWrite(MOTOR_RIGHT_PWM_PIN, 220);
    } else if (leftSensor == 0 && rightSensor == 0) {
        //both sensors are on black
        //move forward
        digitalWrite(MOTOR_LEFT_DIR_PIN, LOW);
        digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
        analogWrite(MOTOR_LEFT_PWM_PIN, 255);
        analogWrite(MOTOR_RIGHT_PWM_PIN, 255);
    } else if (leftSensor == 1 && rightSensor == 0) {
        //right sensor on black and left sensor on white
        //take right turn
        digitalWrite(MOTOR_LEFT_DIR_PIN, LOW);
        digitalWrite(MOTOR_RIGHT_DIR_PIN, HIGH);
        analogWrite(MOTOR_LEFT_PWM_PIN, 220);
        analogWrite(MOTOR_RIGHT_PWM_PIN, 220);
    }
  }
```

- In this code, the number "1" in the while statement: "while (1) {" is used as a boolean value to create an infinite loop. In programming, the number "1" is often used as a shorthand for "true", and in this case, the loop will continue to execute as long as the condition "1" is true.

- This infinite loop will execute the code inside of it repeatedly until the program is stopped or the loop is broken out of. In this case, it reads the sensor values, checks the sensor values and based on that it move the robot forward, left or right.

4) Integrate the weight sensor code (done in part c) to navigate according to the given conditions. The proximity sensor/s (part b) should be used to track down the docking station/unloading bays. The relevant operator will press the push button after unloading the goods at the respective bay.

```
  // Wait for initial button press
  while (!debounce(BUTTON_PIN)) {
    // do nothing
  }

}

void loop();{
  weight = loadCell.get_units();
  if (weight >= 0 && weight <= 3) {
    if (weight < 1.0) {
      followLine();
      checkColor();
      while (!atStationA) {
        // do nothing
      }
      stopRobot();
      // wait for button press
      while (!debounce(BUTTON_PIN)) {
        // do nothing
      }
      // turn around and follow line back to start
      turnAround();
      followLine();
    }
    else if (weight >= 1.0 && weight < 2.0) {
      // robot will stop at station A and B and return
      followLine();
      checkColor();
      while (!atStationA) {
        // do nothing
      }
      stopRobot();
      // wait for button press
      while (!debounce(BUTTON_PIN)) {
        // do nothing
      }
      followLine();
      checkColor();
      while (!atStationB) {
        // do nothing
      }
      stopRobot();
      // wait for button press
      while (!debounce(BUTTON_PIN)) {
        // do nothing
      }
      // turn around and follow line back to start
      turnAround();
      followLine();
    }
    else if (weight >= 2.0 && weight < 3.0) {
     // the robot will stop at all 3 stations A,B and C
      followLine();
      checkColor();
      while (!atStationA) {
        // do nothing
      }
      stopRobot();
      // wait for button press
      while (!debounce(BUTTON_PIN)) {
        // do nothing
      }
      followLine();
      checkColor();
      while (!atStationB) {
        // do nothing
      }
      stopRobot();
      // wait for button press
      while (!debounce(BUTTON_PIN)) {
        // do nothing
      }
      followLine();
      checkColor();
      while (!atStationC) {
        // do nothing
      }
      stopRobot();
      // wait for button press
      while (!debounce(BUTTON_PIN)) {
        // do nothing
      }
      // turn around and follow line back to start
      turnAround();
            followLine();
    }
    else {

      stopRobot();
    }
  }
  else {
    stopRobot();

  }
}  }
```

5) Design and develop the code to return the robot back to the docking station. Highlight design specifications (in robot and line) you will be adhered to make sure it will return. You may use separate diagram to describe how your algorithm works.

To make the robot return I've used the turnaround and follow line command , whereas per turnaround function the robot will make a 180 degree turn and stop after that it'll work under the follow line command where it will follow the black line and it'll stop if it's no more on the black line where the black line starts at docking station

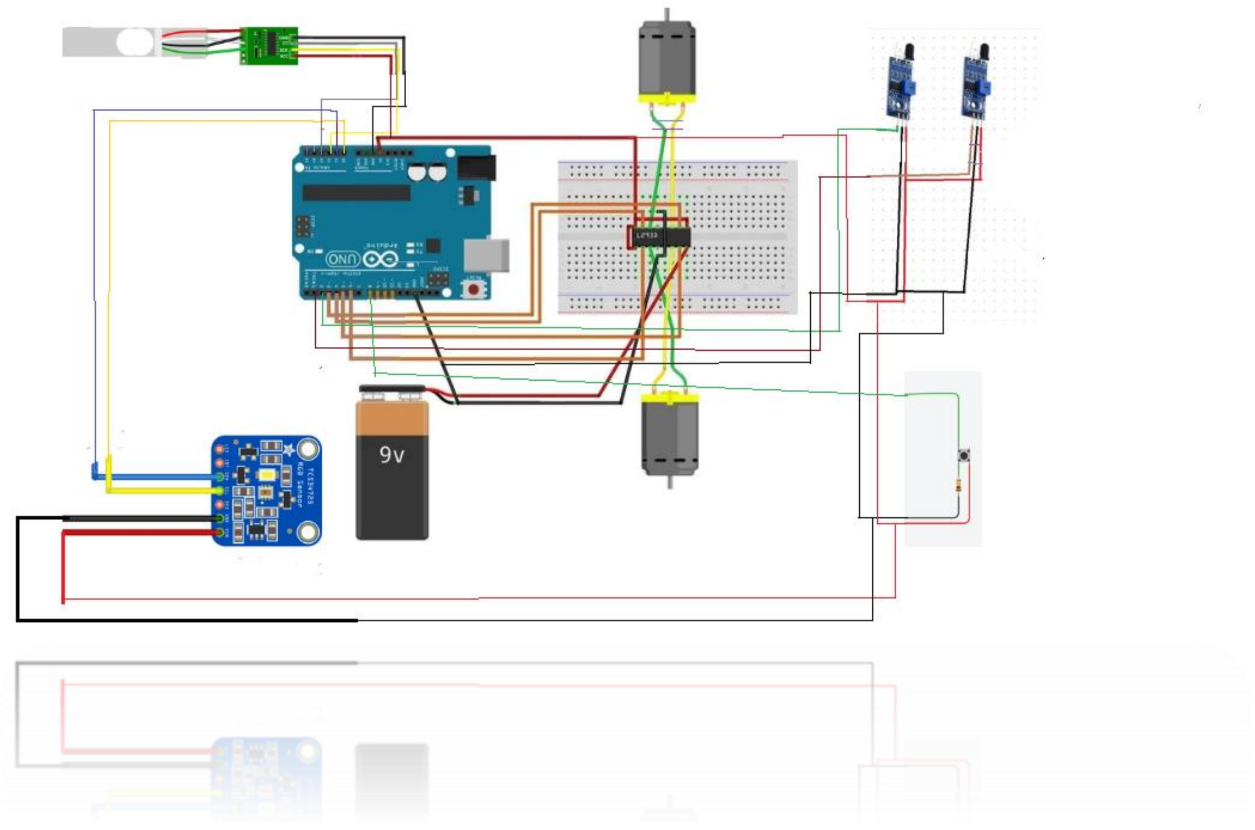
```
void turnAround() {
  // stop the robot
  stopRobot();

  // turn left for 8 seconds
  digitalWrite(MOTOR_LEFT_DIR_PIN, HIGH);
  digitalWrite(MOTOR_RIGHT_DIR_PIN, LOW);
  analogWrite(MOTOR_LEFT_PWM_PIN, 255);
  analogWrite(MOTOR_RIGHT_PWM_PIN, 255);
  delay(8000);  // 8 seconds

  // stop the robot again
  stopRobot();
}
```

- **Also, I have assumed that it will take around 8 second for the robot to make a 180 degree turn**

g) Draw a wiring diagram to show the connections between the micro controller board and other components. The diagram should be very clear to identify the connecting paths.



h) Explore other applications and scenarios where the mobile robots are used in the industry. Find two examples of commercial mobile robots and their usage.

Other examples of applications and scenarios where mobile robots are used in the industry include:

- Agriculture: Mobile robots are being used to perform tasks such as crop monitoring, planting, and harvesting.
- Construction: Mobile robots are being used for tasks such as inspection, measurement, and mapping of construction sites.
- Retail: Mobile robots are being used for tasks such as inventory management, restocking shelves, and providing customer service.
- Mining: Mobile robots are being used for tasks such as exploration, mapping, and monitoring of mining sites.

- Cleaning: Mobile robots are being used for tasks such as floor cleaning, window cleaning and pool cleaning in commercial buildings.
- Security: Mobile robots are being used for tasks such as monitoring and surveillance in public spaces, such as airports and malls.

Examples of commercial mobile robots used in these industries include:

- Harvest CROO Robotics, used for strawberry harvesting.
- Boston Dynamics Spot robot used for site inspections and mapping in construction.
- Focal Systems robots used for inventory management and restocking in retail stores.
- Autonomous Solutions' mining robots used for monitoring and mapping mining sites.
- Avidbots Neo robot used for floor cleaning in commercial buildings.
- Knightscope K5 robot used for security monitoring in public spaces.

Examples of commercial robots and their usage,

1) Mobile robots are widely used in the logistics and warehouse industry for tasks such as inventory management, order picking, and package delivery. One example of a commercial mobile robot used for these tasks is the Kiva robot, which is used by companies like Amazon to automate their warehouses.

2) Mobile robots are also used in healthcare for tasks such as delivering medication and supplies to patients, and assisting with mobility for patients with disabilities. One example of a commercial mobile robot used for these tasks is the TUG robot, which is used in hospitals to transport medication and other supplies throughout the facility.

[3]–[6]

[1] “Arduino UNO Rev3 with Long Pins | Arduino Documentation | Arduino Documentation.” https://docs.arduino.cc/retired/boards/arduino-uno-rev3-with-long-pins (accessed Jan. 16, 2023).

[2] “How To Control A DC Motor With L293D Driver IC Using Arduino - Makerguides.com.” https://www.makerguides.com/how-to-control-a-dc-motor-with-l293d-driver-ic-using-arduino/ (accessed Jan. 17, 2023).

[3] “Insider.” https://www.businessinsider.com/kivas-robots-are-revolutionizing-the-warehouse-industry-2011-4 (accessed Jan. 18, 2023).

[4] “News | Healthcare IT News.” https://www.healthcareitnews.com/news/tug-robot-brings-medication-and-other-supplies-patients-hospitals (accessed Jan. 18, 2023).

[5] “Harvest CROO Robotics.” https://www.harvestcroorobotics.com/ (accessed Jan. 18, 2023).

[6] “Spot® - The Agile Mobile Robot | Boston Dynamics.” https://www.bostondynamics.com/products/spot (accessed Jan. 18, 2023).

Ccc