To accomplish this whole project, you will need to Write code for both the Arduino Mega 2560 and the NodeMCU ESP8266. The Arduino Mega will be responsible for controlling the motors, IR sensor, and LCD display, while the NodeMCU ESP8266 will be responsible for communicating with the Arduino IoT Cloud and sending control signals to the Arduino Mega.

Here's an overview of the steps you can follow to complete this project:

Connect all the components to the Arduino Mega 2560 board, including the IR sensor, limit switches, motor drivers, and LCD display.

you will need to test all components separately, Write a single code for every component you are using then upload to check.

Note: before uploading any code, disconnect **tx** and **rx** wires from the board you are uploading the code on and after uploading you can reconnect them.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the IR sensor:

const int ir\_sensor = 51; //define the pin for the IR sensor

void setup() {

Serial.begin(9600); //initialize the serial communication

pinMode(ir\_sensor, INPUT); //set the pin for the IR sensor as input

}

void loop() {

int ir\_value = digitalRead(ir\_sensor); //read the value from the IR sensor

Serial.println(ir\_value); //print the value to the serial monitor

delay(100); //wait for 100 milliseconds before reading again

}

This code will read the value from the IR sensor connected to pin 51 of the Arduino Mega and print it to the serial monitor. You can use the serial monitor to verify that the sensor is working correctly. When an object is detected by the IR sensor, the value read should be LOW (0), and when no object is detected, the value should be HIGH (1).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the High Limit contact Switch:

const int switch\_pin = 49; //define the pin for the switch

void setup() {

Serial.begin(9600); //initialize the serial communication

pinMode(switch\_pin, INPUT\_PULLUP); //set the pin for the switch as input with internal pull-up resistor

}

void loop() {

int switch\_state = digitalRead(switch\_pin); //read the state of the switch

Serial.println(switch\_state); //print the state to the serial monitor

delay(100); //wait for 100 milliseconds before reading again

}

This code will read the state of a high limit contact switch connected to pin 49 of the Arduino Mega and print it to the serial monitor. You can use the serial monitor to verify that the switch is working correctly. When the switch is closed (limit exceeded), the state read should be LOW (0), and when the switch is open (limit not exceeded), the state should be HIGH (1). Note that INPUT\_PULLUP is used to enable the internal pull-up resistor, which will keep the pin at a HIGH state when the switch is open. If you want to know why I used the pullup , just message me.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the Low Limit contact Switch:

const int switch\_pin = 47; //define the pin for the switch

void setup() {

Serial.begin(9600); //initialize the serial communication

pinMode(switch\_pin, INPUT\_PULLUP); //set the pin for the switch as input with internal pull-up resistor

}

void loop() {

int switch\_state = digitalRead(switch\_pin); //read the state of the switch

Serial.println(switch\_state); //print the state to the serial monitor

delay(100); //wait for 100 milliseconds before reading again

}

This code will read the state of a high limit contact switch connected to pin 47of the Arduino Mega and print it to the serial monitor. You can use the serial monitor to verify that the switch is working correctly. When the switch is closed (limit exceeded), the state read should be LOW (0), and when the switch is open (limit not exceeded), the state should be HIGH (1).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the Liquid crystal display:

Note : if you get an error on the first line or the second , you will need to download the library. If you do not know how to download it, just message me.

#include <Wire.h> //include the Wire library for I2C communication

#include <LiquidCrystal\_I2C.h> //include the LiquidCrystal\_I2C library for LCD control

const int lcd\_columns = 16; //define the number of columns for the LCD

const int lcd\_rows = 2; //define the number of rows for the LCD

LiquidCrystal\_I2C lcd(0x27, lcd\_columns, lcd\_rows); //create an instance of the LiquidCrystal\_I2C class with the I2C address of the LCD and the number of columns and rows

void setup() {

lcd.init(); //initialize the LCD

lcd.backlight(); //turn on the backlight

lcd.setCursor(0, 0); //set the cursor to the first column of the first row

lcd.print("LCD I2C Test"); //print a message to the LCD

}

void loop() {

//do nothing

}

This code will initialize and test an LCD I2C connected to the Arduino Mega using the LiquidCrystal\_I2C library. The LCD is assumed to have an I2C address of 0x27 and a size of 16 columns and 2 rows. The code will turn on the backlight, set the cursor to the first column of the first row, and print a message to the LCD. You can modify the message or add additional LCD control commands as needed. The loop function is empty, so it will simply run the setup code once and then do nothing.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the first motor driver(the first and the second pumps):

const int in1\_pin = 44; //define the pin for IN1 of the second pump

const int in2\_pin = 42; //define the pin for IN2 of the second pump

const int enable1\_pin = 8; //define the pin for the enable of the first pump

const int in3\_pin = 50; //define the pin for IN3 of the first pump

const int in4\_pin = 48; //define the pin for IN4 of the first pump

const int enable2\_pin = 9; //define the pin for the enable of the second pump

void setup() {

pinMode(enable1\_pin, OUTPUT); //set the enable pin for the first pump as output

pinMode(in3\_pin, OUTPUT); //set IN3 for the first pump as output

pinMode(in4\_pin, OUTPUT); //set IN4 for the first pump as output

pinMode(enable2\_pin, OUTPUT); //set the enable pin for the second pump as output

pinMode(in1\_pin, OUTPUT); //set IN1 for the second pump as output

pinMode(in2\_pin, OUTPUT); //set IN2 for the second pump as output

}

void loop() {

//turn on the first pump

analogWrite(enable1\_pin, 255); //enable the first pump in the top speed

digitalWrite(in3\_pin, HIGH); //set IN3 for the first pump as HIGH

digitalWrite(in4\_pin, LOW); //set IN4 for the first pump as LOW

delay(5000); //run the first pump for 5 seconds

digitalWrite(enable1\_pin, LOW); //disable the first pump

//turn on the second pump

analogWrite(enable2\_pin, 255); //enable the second pump in the top speed

digitalWrite(in1\_pin, HIGH); //set IN1 for the second pump as HIGH

digitalWrite(in2\_pin, LOW); //set IN2 for the second pump as LOW

delay(5000); //run the second pump for 5 seconds

digitalWrite(enable2\_pin, LOW); //disable the second pump

//wait for 5 seconds before starting again

delay(5000);

}

This code will turn on two pumps connected to an L298N motor driver module with the following connections:

The first pump: Enable pin to pin 8, IN3 to pin 50, and IN4 to pin 48.

The second pump: Enable pin to pin 9, IN1 to pin 44, and IN2 to pin 42.

The code will turn on each pump for 5 seconds and then turn it off before moving on to the next pump. The loop will repeat indefinitely, with a 5-second delay between each cycle.

Note that you can modify the delay time and the pump control commands as needed to control the pumps according to your specific requirements. Also, be sure to connect the power supply and ground of the L298N module properly, and make sure that the motor voltage and current requirements are within the specifications of the L298N module.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the second motor driver(the third and the fourth pumps):

const int enable1\_pin = 10; //define the pin for the enable of the third pump

const int in3\_pin = 38; //define the pin for IN3 of the third pump

const int in4\_pin = 36; //define the pin for IN4 of the third pump

const int enable2\_pin = 11; //define the pin for the enable of the fourth pump

const int in1\_pin = 32; //define the pin for IN1 of the fourth pump

const int in2\_pin = 30; //define the pin for IN2 of the fourth pump

void setup() {

pinMode(enable1\_pin, OUTPUT); //set the enable pin for the third pump as output

pinMode(in3\_pin, OUTPUT); //set IN3 for the third pump as output

pinMode(in4\_pin, OUTPUT); //set IN4 for the third pump as output

pinMode(enable2\_pin, OUTPUT); //set the enable pin for the fourth pump as output

pinMode(in1\_pin, OUTPUT); //set IN1 for the second pump as output

pinMode(in2\_pin, OUTPUT); //set IN2 for the second pump as output

}

void loop() {

//turn on the first pump

analogWrite(enable1\_pin, 255); //enable the third pump in the top speed

digitalWrite(in3\_pin, HIGH); //set IN3 for the third pump as HIGH

digitalWrite(in4\_pin, LOW); //set IN4 for the third pump as LOW

delay(5000); //run the first pump for 5 seconds

digitalWrite(enable1\_pin, LOW); //disable the first pump

//turn on the second pump

analogWrite(enable2\_pin, 255); //enable the fourth pump in the top speed

digitalWrite(in1\_pin, HIGH); //set IN1 for the fourth pump as HIGH

digitalWrite(in2\_pin, LOW); //set IN2 for the fourth pump as LOW

delay(5000); //run the fourth pump for 5 seconds

digitalWrite(enable2\_pin, LOW); //disable the fourth pump

//wait for 5 seconds before starting again

delay(5000);

}

This code will turn on two pumps connected to an L298N motor driver module with the following connections:

The first pump: Enable pin to pin 10, IN3 to pin 38, and IN4 to pin 36.

The second pump: Enable pin to pin 11, IN1 to pin 32, and IN2 to pin 30.

The code will turn on each pump for 5 seconds and then turn it off before moving on to the next pump. The loop will repeat indefinitely, with a 5-second delay between each cycle.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the third motor driver(the drive motor and the solenoid):

// variables for the Drive Motor

const int DRIVE\_MOTOR\_ENABLE\_PIN = 12;

const int DRIVE\_MOTOR\_IN1\_PIN = 37;

const int DRIVE\_MOTOR\_IN2\_PIN = 35;

// variables for the Solenoid Motor

const int SOLENOID\_MOTOR\_ENABLE\_PIN = 13;

const int SOLENOID\_MOTOR\_IN3\_PIN = 43;

void setup() {

pinMode(DRIVE\_MOTOR\_ENABLE\_PIN, OUTPUT); //set the enable pin for the Drive Motor as output

pinMode(DRIVE\_MOTOR\_IN1\_PIN, OUTPUT); //set IN1 for the Drive Motor as output

pinMode(DRIVE\_MOTOR\_IN2\_PIN, OUTPUT); //set IN2 for the Drive Motor as output

pinMode(SOLENOID\_MOTOR\_ENABLE\_PIN, OUTPUT); //set the enable pin for the Solenoid Motor as output

pinMode(SOLENOID\_MOTOR\_IN3\_PIN, OUTPUT); //set IN3 for the Solenoid Motor as output

digitalWrite(41, LOW); //set pin 41 to ground

}

void loop() {

//turn on the Drive Motor

digitalWrite(DRIVE\_MOTOR\_ENABLE\_PIN, HIGH); //enable the Drive Motor

digitalWrite(DRIVE\_MOTOR\_IN1\_PIN, HIGH); //set IN1 for the Drive Motor as HIGH

digitalWrite(DRIVE\_MOTOR\_IN2\_PIN, LOW); //set IN2 for the Drive Motor as LOW

delay(5000); //run the Drive Motor for 5 seconds

digitalWrite(DRIVE\_MOTOR\_ENABLE\_PIN, LOW); //disable the Drive Motor

//turn on the Solenoid Motor

digitalWrite(SOLENOID\_MOTOR\_ENABLE\_PIN, HIGH); //enable the Solenoid Motor

digitalWrite(SOLENOID\_MOTOR\_IN3\_PIN, HIGH); //set IN3 for the Solenoid Motor as HIGH

delay(5000); //run the Solenoid Motor for 5 seconds

digitalWrite(SOLENOID\_MOTOR\_ENABLE\_PIN, LOW); //disable the Solenoid Motor

//wait for 5 seconds before starting again

delay(5000);

}

This code will turn on two motors connected to an L298N motor driver module with the following connections:

The Drive Motor: Enable pin to pin 12, IN1 to pin 37, and IN2 to pin 35.

The Solenoid Motor: Enable pin to pin 13, and IN3 to pin 43.

The code will turn on each motor for 5 seconds and then turn it off before moving on to the next motor. The Drive Motor will run in one direction, and the Solenoid Motor will run in the opposite direction. Pin 41 is set to ground. The loop will repeat indefinitely, with a 5-second delay between each cycle.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here are two codes to test the serial (UART) communication between the two boards:

Upload the code to the two boards and connect the Tx and Rx pins and after that you must open the serial monitor for both the two boards and see the message.

NodeMCU code:

void setup() {

// Start serial communication

Serial.begin(9600);

}

void loop() {

// Send a message to the Arduino Mega every second

Serial.println("Hello Arduino Mega!");

delay(1000);

}

Arduino Mega code:

void setup() {

// Start serial communication

Serial.begin(9600);

}

void loop() {

// Check if a message has been received from the NodeMCU

if (Serial.available() > 0) {

// Read the message

String message = Serial.readStringUntil('\n');

// Print the message to the serial monitor

Serial.println(message);

}

}

The NodeMCU code will send a message to the Arduino Mega every second using the hardware serial communication. The Arduino Mega code will continuously check if there is a message available from the NodeMCU, and if so, it will read the message and print it to the serial monitor. Note that the baud rate is set to 9600 in both codes, so make sure that the serial monitor of each device is also set to 9600 baud. Also, make sure that the NodeMCU and the Arduino Mega are connected through their respective RX and TX pins. Which means that Tx pin of the Arduino mega must be connected to the Rx of the nodeMCU and vice versa.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a code for the IR receiver:

Note : if you get an error on the first line , you will need to download the IRremote library. If you do not know how to download it, just message me.

#include <IRremote.h>

const int ir\_receiver = 53; //define the pin for the IR receiver

IRrecv irrecv(ir\_receiver); //create an instance of the IRrecv class

decode\_results results; //create a variable to store the IR code results

void setup() {

Serial.begin(9600); //initialize the serial communication

irrecv.enableIRIn(); //enable the IR receiver

}

void loop() {

if (irrecv.decode(&results)) { //check if an IR code is received

Serial.println(results.value, HEX); //print the IR code to the serial monitor in hexadecimal format

irrecv.resume(); //reset the IR receiver for the next code

}

}

This code will read the IR codes received by an IR receiver connected to pin 53 of the Arduino Mega and print them to the serial monitor in hexadecimal format. You can use an IR remote to send IR codes to the receiver and verify that the receiver is working correctly. When an IR code is received, the code will be printed to the serial monitor.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is the code to test your remote :

After getting the 4 hex codes you have sent.

#include <IRremote.h>

#define IR\_RECEIVER 53

// IR remote setup

IRrecv irrecv(IR\_RECEIVER);

decode\_results results;

unsigned long button1 = 0xFF30CF;

unsigned long button2 = 0xFF18E7;

unsigned long button3 = 0xFF7A85;

unsigned long button4 = 0xFF10EF;

void setup() {

// Initialize serial communication

Serial.begin(9600);

// Initialize IR remote

irrecv.enableIRIn();

}

void loop() {

if (irrecv.decode(&results)) {

if (results.value == button1) {

Serial.println("button 1 is pressed");

}

else if (results.value == button2) {

Serial.println("button 2 is pressed");

}

else if (results.value == button3) {

Serial.println("button 3 is pressed");

}

else if (results.value == button4) {

Serial.println("button 4 is pressed");

}

irrecv.resume();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

After testing all components, now you are sure that there aren’t any errors in any component or its connection.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now you must test the Wi-Fi connection and the IOT cloud:

First you must write your own SSID and password in arduino\_secret.h file which will appears as a tab on the top beside the code then upload the code of the nodeMCU and turn on the serial Monitor and push on each pushbutton on the application you are logged in using this email: [peepless755@gmail.com](mailto:peepless755@gmail.com)

And this password: Steven peepless755 and then see the number on the serial monitor. Do not forget to check if the board is online or offline on the website (Arduino IOT cloud).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Then you must open Amazon Alexa mobile application using your account and go to skills and search for Arduino skill then enable it and choose the devices you must link which are the switches or the pushbuttons , then say “Alexa, turn on pushbutton 1 and test the voice control and change has done on the other side or not If it has change and everything is ok you must change turn on pushbutton 1 to the needed command which is dispense drink 1 , tea or whatever you want How can you do that??!! Go to the next section.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

It is easy Mr. Steven Just

1. Open the Alexa app and tap on "Routines" in the menu.

2. Tap on the "+" icon in the top right corner to create a new routine.

3. Give your routine any name, such as "Turn on Pushbutton one."

4. Tap on "When this happens" and select "Voice" as the trigger. Enter a custom phrase such as "Alexa, dispense coffee" or "any phrase you want.”

5. Tap on "Add action" and select "Smart Home" as the action type.

6. Select the switch or the pushbutton that you want to turn off, such as "pushbutton1."

7. Select "Turn on" as the action.

8. Tap on "Save" to save your routine.

Now, when you say your custom phrase, Alexa will turn on the pushbutton. You can also create a similar routine to turn the other pushbuttons using a custom phrase of your choice.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Finally, Write the whole code for the Arduino Mega 2560 board. This code should include functions to control the motors, read the IR sensor and limit switches, and display messages on the LCD display. The code should also include functions to receive input signals from the NodeMCU ESP8266 board.