Cosmothon 2025

KazRockets: Design of Modular Rover for Educational Purposes

Pirog Almatau:

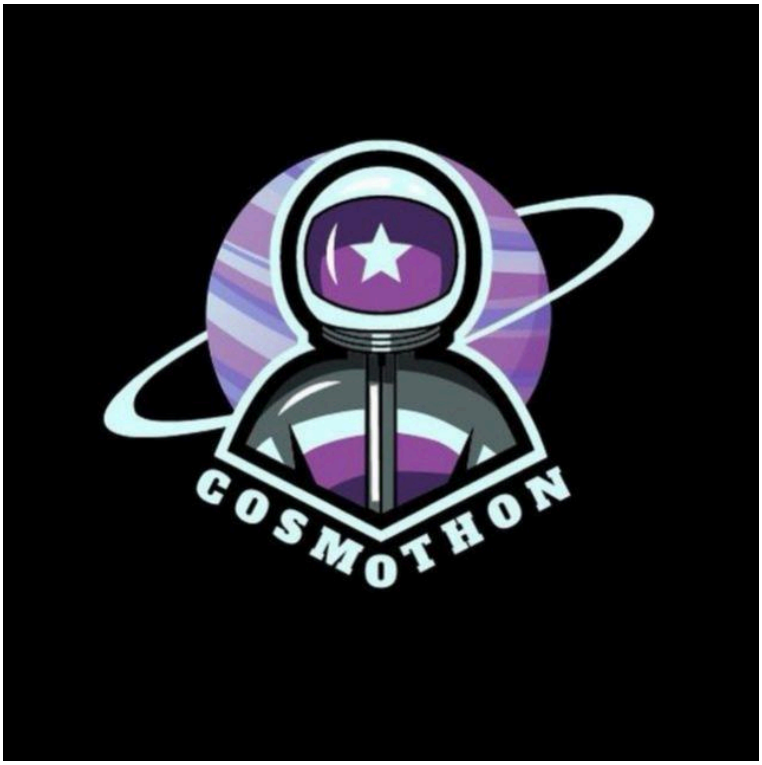
Oleg Gi

Ilyas Kurpetayev

Bektaiyr Tuleshov

Alisher Atrauov

Naoya Matsuda



1.0 Introduction and Model Overview

As society becomes increasingly information-oriented, the importance of acquiring knowledge about electronic devices, the Internet, and information and communication technologies in school education is increasing. This trend is expected to accelerate in the future, and programming skills, in particular, will become the minimum knowledge required for human beings, next to reading and writing. In this context, it is necessary to get students interested in information science at the school education stage.

For this purpose, we would like to propose this learning kit for education. The rover, which is currently the subject of much research and development in the field of space development, will not only stimulate students' interest in space, but also help them learn about electronic circuits, programming, and robotics through the creation of this learning kit. Furthermore, this learning kit is not just assembled according to the instructions, but also allows students to extend the robot's functionality by adding additional attachments to the robot themselves. This is expected to stimulate students' imagination and improve their independent exploration skills.

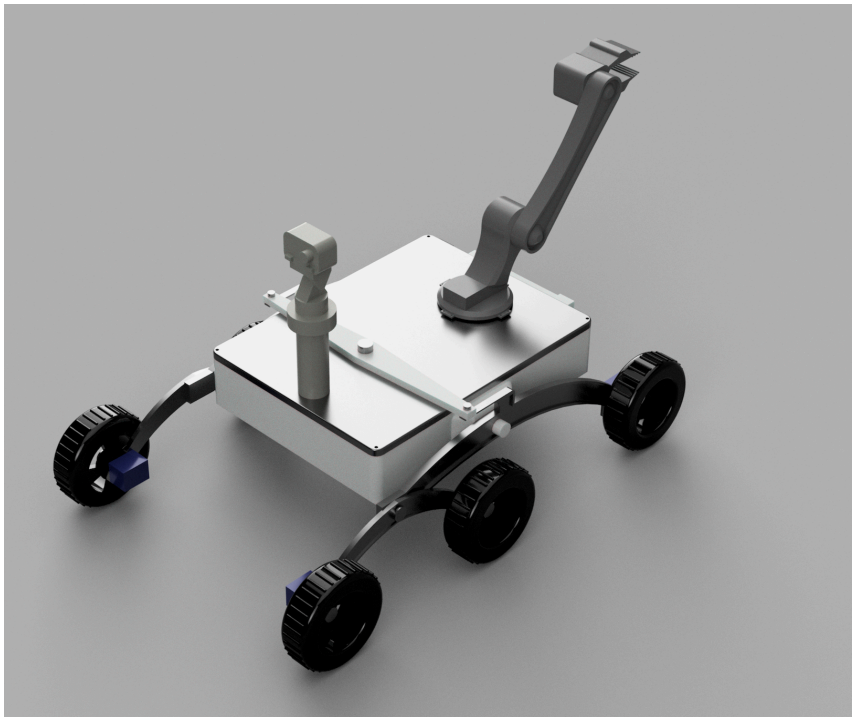


Figure 1. Isometric view of a rover

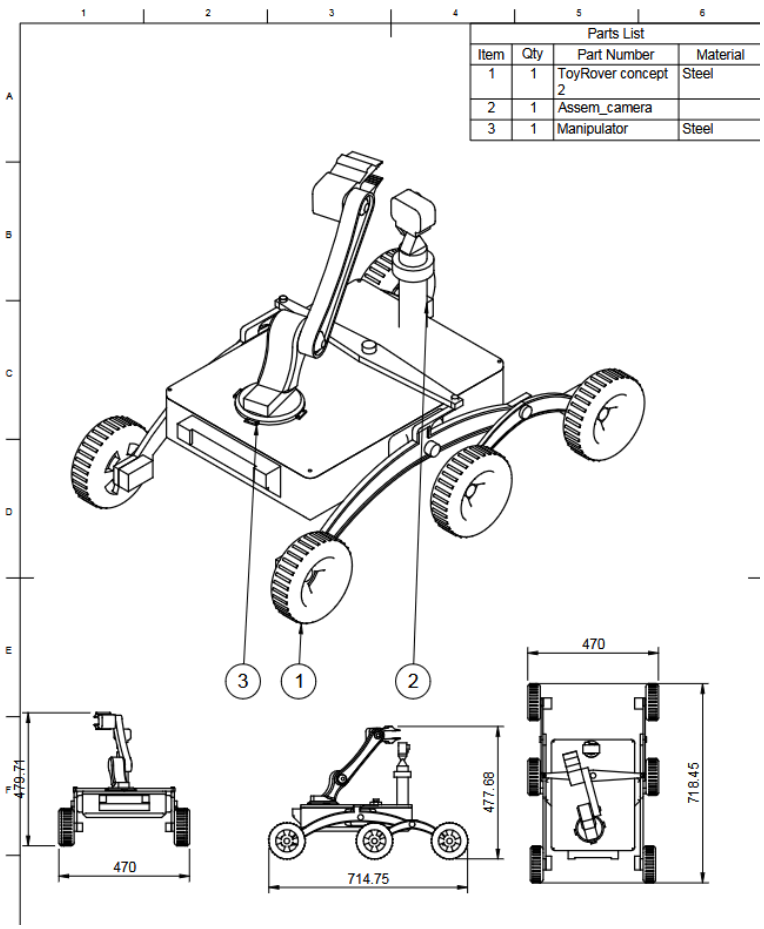


Figure 2. Dimensions of the rover

Length: 718.45 mm (can be reduced with 4 wheel structure instead)

Width: 470 mm

Height: 477.68 mm

The construction was heavily inspired by the Curiosity rover of NASA, which landed on Mars on 6th of August, 2012. The rover had become one of the most well known scientific breakthroughs in history because of its technical capabilities and long lasting mission. As of now, the rover is still operating and is primarily used to collect various geological and biological data from the surface of Mars.

Figure 3. Curiosity Mars Rover, Self-Portrait in October 2015 (source: NASA)

2.0 Construction, Modules and Sensors

The main hull of the rover will be constructed with ABS plastic due to light weight and low cost. Other removable modules can also be made with ABS plastic, since the material also has excellent mechanical properties that will make said parts rigid enough to withstand daily use and additional stresses incurred by traversing through rough surfaces. ABS plastic is also known to have some hydrophobic properties, which will help with keeping the internal parts dry.

Wires should mostly be hidden inside of the structure to minimize risks of breaking or accidents. The only exceptions are modules or sensors that must be attached to the outside. Additional protective layers for said modules and sensors must be used to maximize the safety.

Table 1. List of internal and external modules

Internal parts	**External parts**
Arduino MEGA	Ultrasonic Sensor - HC-SR04
BreadBoard	DHT22/11 Humidity and Temperature Sensor
L298N Motor Driver Board (Qty. 2)	LDR (Mini Photocell)
Standard Gearmotor – 107 RPM (12V) (Qty. 4)	Joysticks and gamepads (optional)
Rechargeable Battery (12V)	Manipulator arm
ESP8266-01 - Wifi Module	Camera
Logic Level Converter - Bi-Directional	4 wheel module
SparkFun ITG-3200 - Triple-Axis Digital-Output Gyro Breakout	Track module
10K Ohm Resistor (Qty. 2)	

Table 2. Total price calculation (assumed to be bought in bulk, package not included)

Part name	**Price per unit, KZT**	**Quantity per model**	**Total price, KZT**
Arduino MEGA	2500	1	2500
BreadBoard	460	1	460
L298N Motor Driver Board	600	2	1200
Standard Gear motor – 107 RPM (12V)	350	4	1400
Rechargeable Battery (12V)	1800	1	1800
ESP8266-01 - Wifi Module	450	1	450
Logic Level Converter - Bi-Directional	125	2	250
SparkFun ITG-3200 - Triple-Axis Digital-Output Gyro Breakout	530	1	530
10K Ohm Resistor	2.5	2	5
Ultrasonic Sensor - HC-SR04	250	1	250

DHT22/11 Humidity and Temperature Sensor	500	1	500
LDR (Mini Photocell)	10	1	10
Joysticks and gamepads	400	1	400
Camera	440	1	440
TOTAL PRICE, KZT	**10195**		

Selecting Arduino and modular construction also has a row of advantages. Namely, the rover will be open to modifications by users themselves and broken parts can be easily purchased and replaced by the users on their own.

As for the general controls of the rover, ESP8266-01 - Wifi Module can be used to connect a smartphone to the rover and some open source projects can be used to control the movements. This will require the connection to the same network, however.

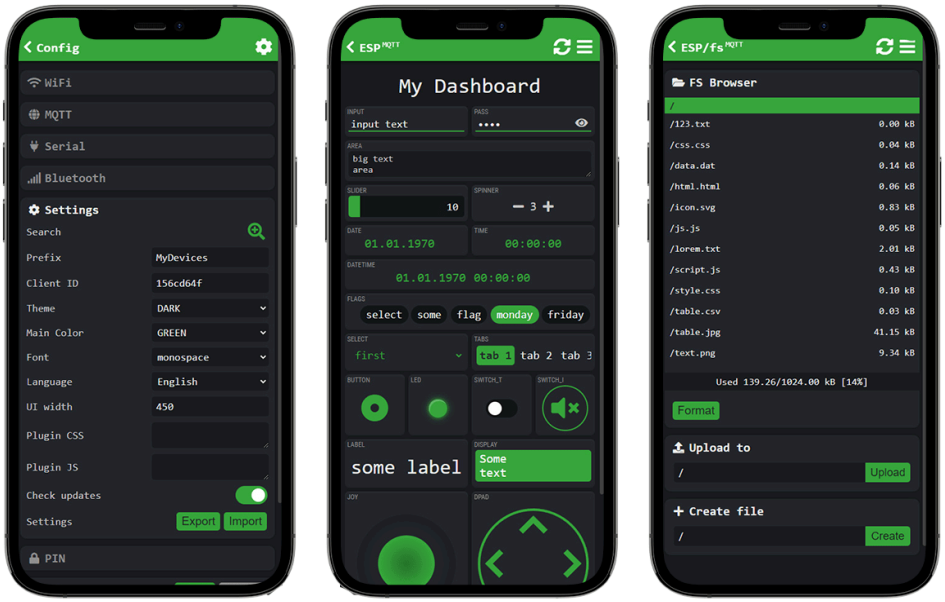


Figure 4. A control panel developed by AlexGyver on GitHub (https://github.com/GyverLibs/GyverHUB?tab=readme-ov-file)

Ultrasonic Sensor - HC-SR04 is an alternative method to control the rover. Arduino's sensor can read distance based on ultrasonic pulses sent by it. Hence, it is possible to create an autonomous control, which will adjust the rover's movements based on the sensor's distance readings. If an obstacle is detected in front, the rover should start to rotate until there is no more obstacles detected ahead.

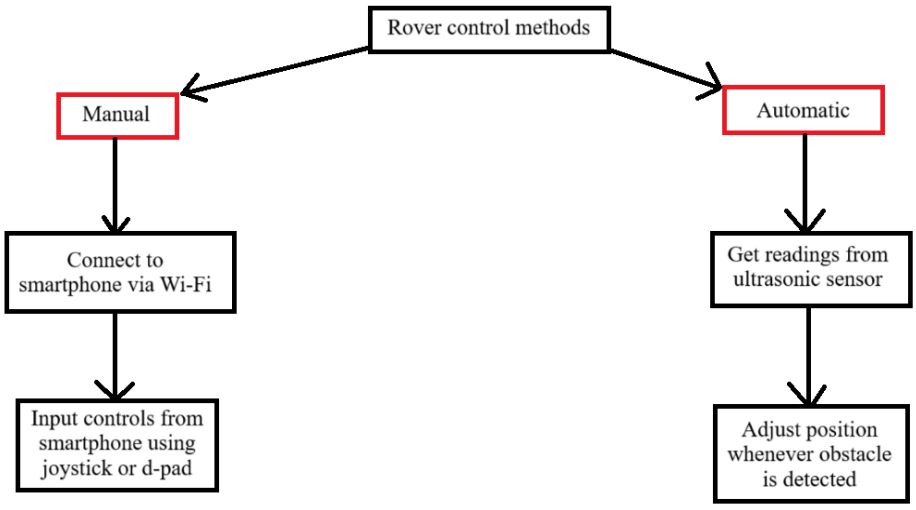


Figure 5. Control methods

As for the wheel modules, the team offers 2 solutions: 6 wheel drive and a a few variations of track modules.

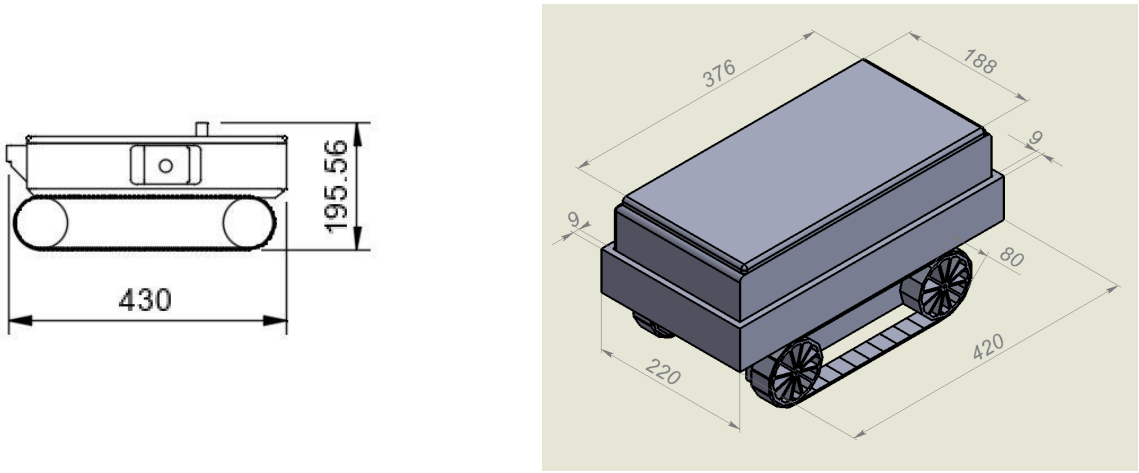


Figure 6. Track modules

3.0 Connections and Sensor Settings

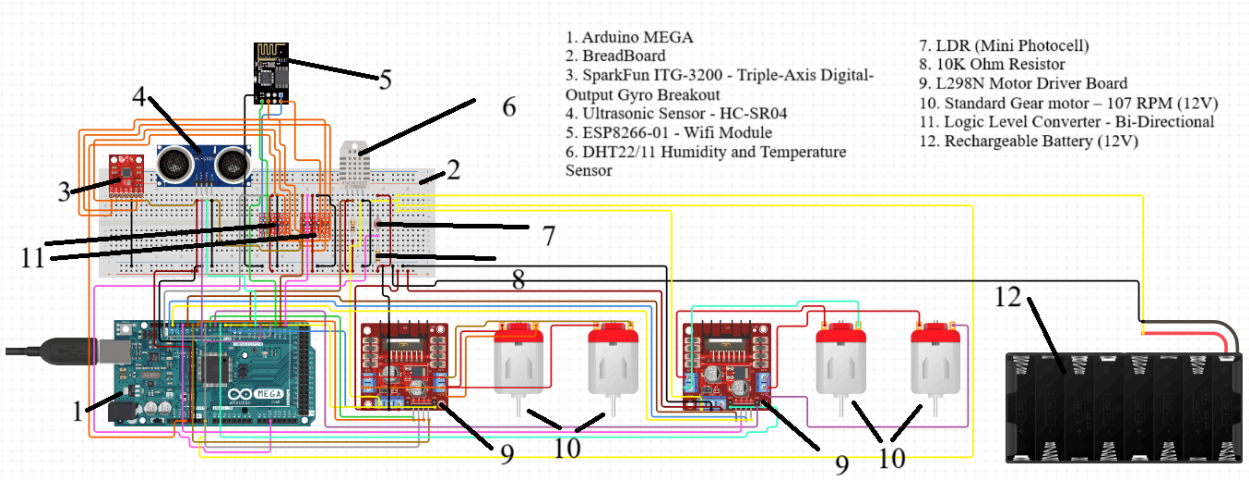


Figure 7. Example of circuit (open to modifications, made with circuito.io)

One of the goals of the project is to familiarize children with programming. While it is optional and our team suggests that if the project is to be distributed, it should come with a preemptively programmed Arduino board, an option and instructions to modify the code will be provided to the users within the package.

Arduino is programmed primarily with Arduino IDE, which may be difficult to understand for children, since it is similar to one of the most difficult coding languages, C++. While it is still an option and guidelines for Arduino IDE can still be included in the packaging, the team also decided to use alternative programming methods that use open source solutions.

One of such methods is to use a table, where pin connections, trigger conditions, and various settings can be adjusted without writing any lines of code.

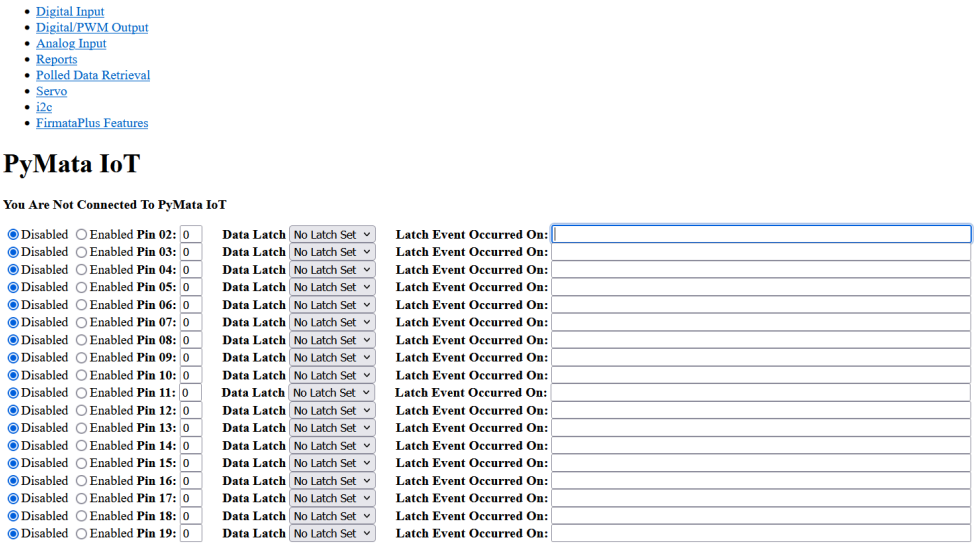


Figure 8. PyMata IoT Tester, an open source project from GitHub by Mr. Y (https://mryslab.github.io/pymata-aio/examples/uno_iot_tester.html)

Another method is to use coding blocks, which replace actual programming with simple blocks that contain commands and some adjustable parameters.

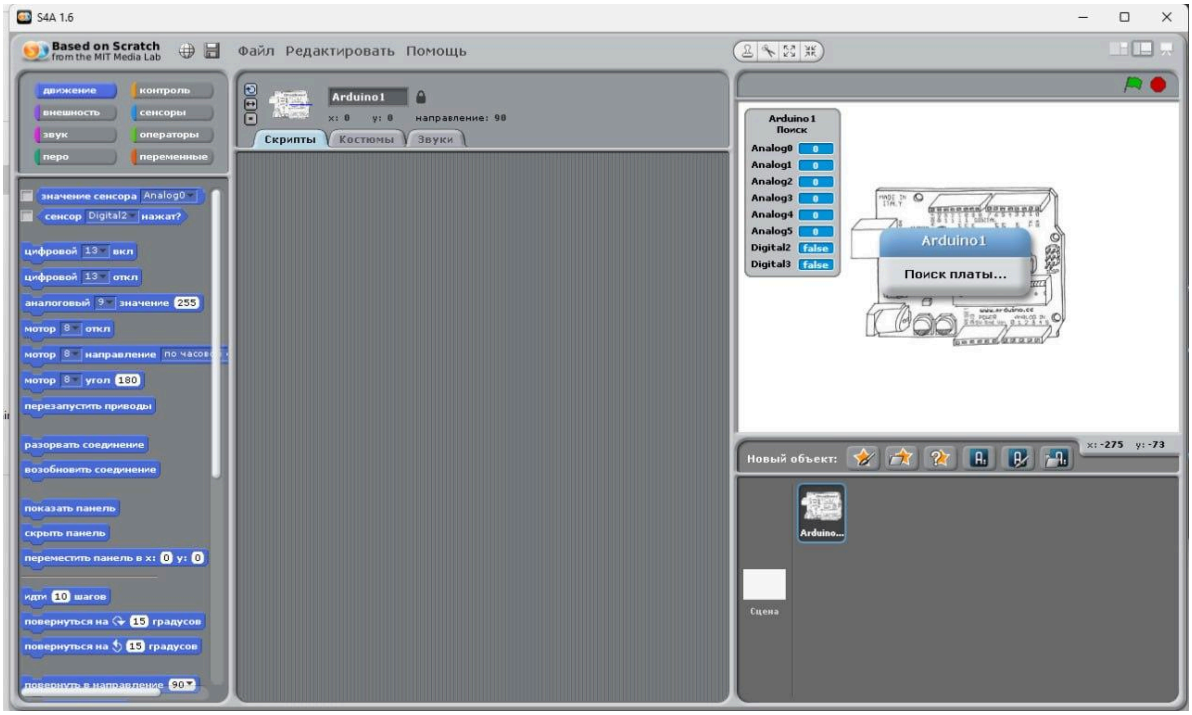


Figure 9. S4A, programming blocks based on Scratch from MIT Media Lab (https://github.com/technologiescollege/BlocklyArduino_AIO?tab=readme-ov-file)

4.0 Conclusion

The final design has the following dimensions: length of 718.45* mm width of 470 mm and height of 477.68 mm (length can be reduced easily). The design was inspired by NASA's Curiosity Mars Rover. The total cost of the project slightly exceeds 10000 KZT, not including the package. The controls of the rover were realized via ESP8266-01 - Wifi Module and connection of both the smartphone and the rover to a common network. The smartphone in this case acts as a controller. The electronic components are connected via Arduino MEGA, which is pre-coded with basic commands. As an extension, the project offers 3 main ways to interact with the code for additional modifications of the rover: via standard Arduino IDE, using tables to create a code based on user inputs and using blocks instead of writing actual lines of code. Most of the components are hidden inside of the hull, while some sensors and external modules are enhanced with additional protective layers.

5.0 Source Code

```
// Include Libraries
#include <Arduino.h>
#include <DCMDriverL298.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Gyro.h>
#include <NewPing.h>
#include <LDR.h>

// Pin Definitions
#define DCMOTORDRIVERL298_1_PIN_INT1 6
#define DCMOTORDRIVERL298_1_PIN_ENB 3
#define DCMOTORDRIVERL298_1_PIN_INT2 7
#define DCMOTORDRIVERL298_1_PIN_ENA 2
#define DCMOTORDRIVERL298_1_PIN_INT3 8
#define DCMOTORDRIVERL298_1_PIN_INT4 9
#define DCMOTORDRIVERL298_2_PIN_INT1 10
#define DCMOTORDRIVERL298_2_PIN_ENB 5
#define DCMOTORDRIVERL298_2_PIN_INT2 11
#define DCMOTORDRIVERL298_2_PIN_ENA 4
#define DCMOTORDRIVERL298_2_PIN_INT3 12
#define DCMOTORDRIVERL298_2_PIN_INT4 13
#define DHT_PIN_DATA 14
#define HCSR04_PIN_TRIG 16
#define HCSR04_PIN_ECHO 15
#define LDR_PIN_SIG A10 // Ensure your board supports A10

// Wi-Fi Credentials
const char* SSID     = "WIFI-SSID";   // Enter your Wi-Fi name
```

```
const char* PASSWORD = "PASSWORD";    // Enter your Wi-Fi password

const char* host     = "www.google.com"; // Corrected host declaration
const int hostPort   = 80;
#define THRESHOLD_LDR   100
int ldrAverageLight;

// Object Instantiations
DCMDriverL298 dcMotorDriverL298_1(
  DCMOTORDRIVERL298_1_PIN_ENA,
  DCMOTORDRIVERL298_1_PIN_INT1,
  DCMOTORDRIVERL298_1_PIN_INT2,
  DCMOTORDRIVERL298_1_PIN_ENB,
  DCMOTORDRIVERL298_1_PIN_INT3,
  DCMOTORDRIVERL298_1_PIN_INT4
);

DCMDriverL298 dcMotorDriverL298_2(
  DCMOTORDRIVERL298_2_PIN_ENA,
  DCMOTORDRIVERL298_2_PIN_INT1,
  DCMOTORDRIVERL298_2_PIN_INT2,
  DCMOTORDRIVERL298_2_PIN_ENB,
  DCMOTORDRIVERL298_2_PIN_INT3,
  DCMOTORDRIVERL298_2_PIN_INT4
);

DHT dht(DHT_PIN_DATA, DHT22); // Specify sensor type
Gyro gyro;
NewPing hcsr04(HCSR04_PIN_TRIG, HCSR04_PIN_ECHO);
LDR ldr(LDR_PIN_SIG);

const unsigned long timeout      = 10000;
char menuOption                  = 0;
unsigned long time0              = 0;

void setup()
{
   Serial.begin(9600);

   #if defined(ARDUINO_SAM_DUE) // Modify condition based on your board
      while (!Serial);
   #endif

   Serial.println("Start");

   dht.begin();

   WiFi.begin(SSID, PASSWORD);
   Serial.print("Connecting to WiFi");
   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }
   Serial.println(" connected");

   ldrAverageLight = ldr.readAverage();
```

```
    time0 = millis();
    menuOption = menu();
}

void loop()
{
    switch(menuOption) {
        case '1':
            dcMotorDriverL298_1.setMotorA(200, 1);
            dcMotorDriverL298_1.setMotorB(200, 0);
            delay(2000);
            dcMotorDriverL298_1.stopMotors();
            delay(2000);
            break;

        case '2':
            dcMotorDriverL298_2.setMotorA(200, 1);
            dcMotorDriverL298_2.setMotorB(200, 0);
            delay(2000);
            dcMotorDriverL298_2.stopMotors();
            delay(2000);
            break;

        case '3': {
            float dhtHumidity = dht.readHumidity();
            float dhtTempC    = dht.readTemperature(); // Updated method name
            Serial.print(F("Humidity: ")); Serial.print(dhtHumidity); Serial.print(F(" [%]\t"));
            Serial.print(F("Temp: ")); Serial.print(dhtTempC); Serial.println(F(" [C]"));
            break;
        }

        case '4': {
            WiFiClient client;
            if (!client.connect(host, hostPort)) {
                Serial.println("Connection failed.");
                break;
            }

            client.println("GET / HTTP/1.1");
            client.print("Host: "); client.println(host);
            client.println("Connection: close");
            client.println();

            unsigned long responseStart = millis();
            while (client.available() == 0) {
                if (millis() - responseStart > 5000) { // 5 seconds timeout
                    Serial.println(">>> Client Timeout !");
                    client.stop();
                    break;
                }
            }

            bool dateFound = false;
            while (client.available()) {
                String line = client.readStringUntil('\n');
```

```
            if (line.startsWith("Date:")) {
                Serial.println(line);
                dateFound = true;
                break;
            }
        }

        if (!dateFound) {
            Serial.println("Date not found in response.");
        }

        client.stop();
        break;
    }

    case '5': {
        int gyroX = gyro.getX();
        int gyroY = gyro.getY();
        int gyroZ = gyro.getZ();
        Serial.print(F("X: ")); Serial.print(gyroX);
        Serial.print(F("\tY: ")); Serial.print(gyroY);
        Serial.print(F("\tZ: ")); Serial.println(gyroZ);
        break;
    }

    case '6': {
        int hcsr04Dist = hcsr04.ping_cm();
        Serial.print(F("Distance: ")); Serial.print(hcsr04Dist); Serial.println(F(" [cm]"));
        break;
    }

    case '7': {
        int ldrSample = ldr.read();
        int ldrDiff   = abs(ldrAverageLight - ldrSample);
        Serial.print(F("Light Diff: ")); Serial.println(ldrDiff);
        break;
    }

    default:
        break;
  }


  if (millis() - time0 > timeout)
  {
    menuOption = menu();
  }
}

char menu()
{
  Serial.println(F("\nWhich component would you like to test?"));
  Serial.println(F("(1) L298N Motor Driver with Dual Standard DC Motors (Geared) #1"));
  Serial.println(F("(2) L298N Motor Driver with Dual Standard DC Motors (Geared) #2"));
  Serial.println(F("(3) DHT22/11 Humidity and Temperature Sensor"));
  Serial.println(F("(4) Logic Level Converter - Bi-Directional"));
```

```
    Serial.println(F("(5) SparkFun ITG-3200 - Triple-Axis Digital-Output Gyro Breakout"));
    Serial.println(F("(6) Ultrasonic Sensor - HC-SR04"));
    Serial.println(F("(7) LDR (Mini Photocell)"));
    Serial.println(F("(menu) send anything else or press on board reset button\n"));

    unsigned long startTime = millis();
    const unsigned long inputTimeout = 10000; // 10 seconds

    while (millis() - startTime < inputTimeout) {
      if (Serial.available()) {
        char c = Serial.read();

        while (Serial.available()) {
          Serial.read();
        }

        if (isAlphaNumeric(c))
        {
          switch(c) {
            case '1':
              Serial.println(F("Now Testing L298N Motor Driver with Dual Standard DC Motors (Geared)
#1"));
              break;
            case '2':
              Serial.println(F("Now Testing L298N Motor Driver with Dual Standard DC Motors (Geared)
#2"));
              break;
            case '3':
              Serial.println(F("Now Testing DHT22/11 Humidity and Temperature Sensor"));
              break;
            case '4':
              Serial.println(F("Now Testing Logic Level Converter - Bi-Directional"));
              break;
            case '5':
              Serial.println(F("Now Testing SparkFun ITG-3200 - Triple-Axis Digital-Output Gyro
Breakout"));
              break;
            case '6':
              Serial.println(F("Now Testing Ultrasonic Sensor - HC-SR04"));
              break;
            case '7':
              Serial.println(F("Now Testing LDR (Mini Photocell)"));
              break;
            default:
              Serial.println(F("Illegal input!"));
              return 0;
          }
          time0 = millis(); // Reset timeout counter
          return c;
        }
        else {
          Serial.println(F("Illegal input!"));
          return 0;
        }
      }
    }
```

```
    Serial.println(F("Input timeout. Returning to main loop."));
    return 0;
}


/*********************************************************

*       Circuito.io is an automatic generator of schematics and code for off
*       the shelf hardware combinations.

*       Copyright (C) 2016 Roboplan Technologies Ltd.

*       This program is free software: you can redistribute it and/or modify
*       it under the terms of the GNU General Public License as published by
*       the Free Software Foundation, either version 3 of the License, or
*       (at your option) any later version.

*       This program is distributed in the hope that it will be useful,
*       but WITHOUT ANY WARRANTY; without even the implied warranty of
*       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
*       GNU General Public License for more details.

*       You should have received a copy of the GNU General Public License
*       along with this program.  If not, see <http://www.gnu.org/licenses/>.

*       In addition, and without limitation, to the disclaimers of warranties
*       stated above and in the GNU General Public License version 3 (or any
*       later version), Roboplan Technologies Ltd. ("Roboplan") offers this
*       program subject to the following warranty disclaimers and by using
*       this program you acknowledge and agree to the following:
*       THIS PROGRAM IS PROVIDED ON AN "AS IS" AND "AS AVAILABLE" BASIS, AND
*       WITHOUT WARRANTIES OF ANY KIND EITHER EXPRESS OR IMPLIED.  ROBOPLAN
*       HEREBY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
*       NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, TITLE, FITNESS
*       FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, AND THOSE ARISING BY
*       STATUTE OR FROM A COURSE OF DEALING OR USAGE OF TRADE.
*       YOUR RELIANCE ON, OR USE OF THIS PROGRAM IS AT YOUR SOLE RISK.
*       ROBOPLAN DOES NOT GUARANTEE THAT THE PROGRAM WILL BE FREE OF, OR NOT
```