Dokuz Eylül University

Faculty of Engineering

Electrical and Electronics

Engineering Department

2023-2024

ETE 3007 (Fundamentals of Robotics)

Course Project

RC Bicycle Robot Construction

Final Report

2020502039 **Zehra Miray GÜLSEREN**

2020502059 **Eda SEZEN**

2020510091 **Evrim Gizem İŞCİ**

2021510081 **Özgün ÖZER**

Submitted Date: 27.05.2024

Course Instructor:

Assoc. Prof. Ahmet ÖZKURT

CONTENTS 2

ABSTRACT

In a world where technology is shaping the way we move, our project aims to bring a fresh perspective to personal mobility. At the heart of our endeavor is the development of a bicycle robot—an innovative blend of robotics, sensor tech, and wireless connectivity.

Our project is driven by a simple yet powerful vision: to create a bicycle that goes beyond traditional limitations. By leveraging cutting-edge advancements, we're crafting a vehicle that seamlessly integrates into modern urban life, offering enhanced convenience and control.

With a focus on simplicity and accessibility, our bicycle robot will revolutionize how people navigate their surroundings. Through careful design and the use of user-friendly technology, we're striving to redefine the concept of personal mobility.

As we embark on this journey, our goal remains crystal clear: to pioneer a new era of urban transportation solutions that blur the lines between man and machine. By embracing innovation and quality, we're shaping a future where exploration knows no bounds and mobility becomes an effortless extension of everyday life.

INTRODUCTION

A remote-controlled device refers to any electronic or mechanical system that can be operated or controlled from a distance, typically using a remote control unit or a wireless interface. These devices leverage various technologies to enable remote operation, allowing users to manipulate their functions, settings, or movements without direct physical contact.

Remote control connection can be made using the internet, Bluetooth, infrared (IR) light, and radio signals. It can be used for devices like TVs, house appliances, cars, drones, vehicles, etc. Remote control's main purpose is to make life easier.

Remote-controlled vehicles are used for scientific purposes, space probes, submarines, law enforcement, military engagements, and entertainment. Remote-controlled bicycles are one of them and are mostly used for entertainment.

Bicycles have long been a popular mode of transportation and recreation, offering benefits such as eco-friendliness, health promotion, and cost-effectiveness. However, traditional bicycles rely solely on manual input from the rider, limiting their potential for automation and remote operation. With the advent of IoT (Internet of Things) technologies and microcontrollers, bicycles can be enhanced with remote control capabilities, thereby opening up new possibilities for convenience, safety, and functionality.

In a remote-controlled bicycle robot, the most important part is maintaining balance. It can be provided by constructing the bike in a symmetrical form, using sensors to understand the tilt of the bicycle and good steering control.

FEASIBILITY STUDY

To make this project there are three basic parts we need to do. The balance provider, velocity supply and remote control.

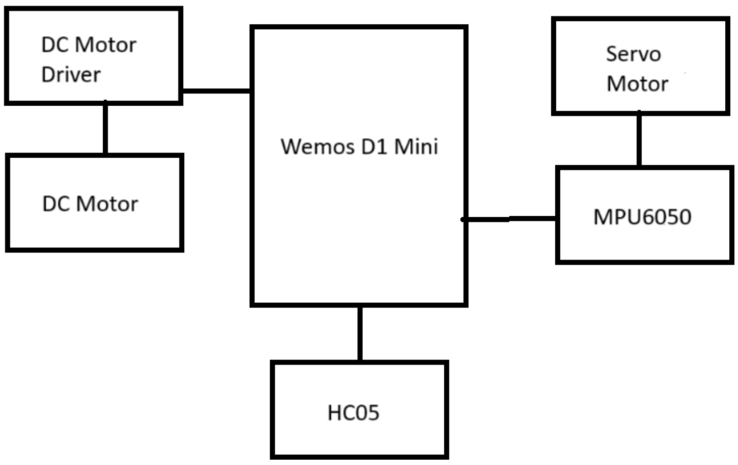
We will provide the balance by using a Gyro Sensor. We decided to use MPU6050. This sensor will determine the position of the bicycle. By connecting this sensor to a servo motor, we can provide the motion of the steering wheel. We need to place the MPU6050 that it lies the flat on the chassis of the bicycle, with this way it can find the data in the balance position.

For the second part we need to use a DC motor to provide the motion of the bicycle. This motor will just connected to the back wheel. The front wheel for the direction and the back one is for the power supply. We will connect the DC motor to the back wheel and we need to connect a driver card for it to work. For not to prevent the balance of the bicycle we need to choose as small as possible the DC motor and driver.

For the remote control we decided to use HC05 Bluetooth Serial Module Card. It provide remote control from the mobile device.

To make all of them work we need to connect all of the parts to a microcontroller. We decided to use Wemos D1 Mini which is also called as ESP32.

Here the schema of the circuit design of the project is given.



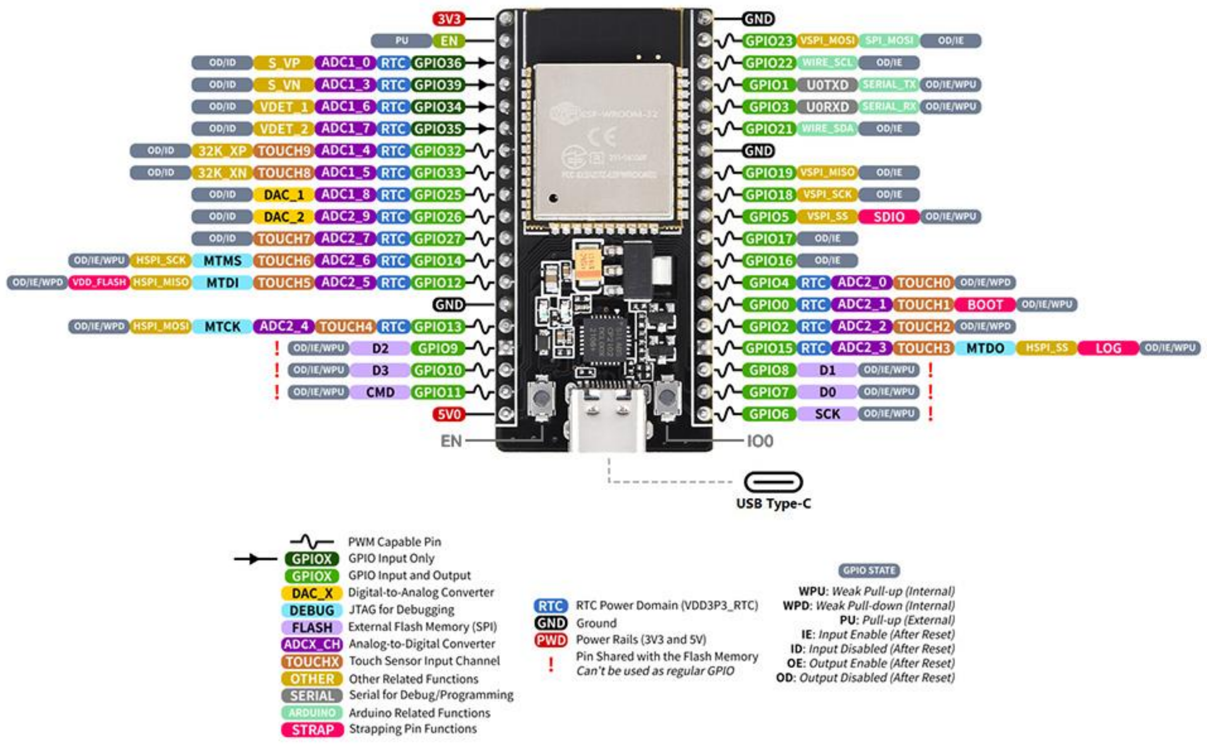
Schema 1 : Schema of the Self Balanced Bicycle Robot

APPLICATION

Introducing the Components

ESP32

We have used NodeMCU development board based on ESP32, ESP-WROOM-32, as the microcontroller of the circuit. ESP32 based modules have integrated flash memories and hence provide effective solutions for Wi-Fi and Bluetooth based connectivity applications[1]. They can also be configured for different applications such as smart home applications, industrial automation, wearable devices, IP camera and intelligent agriculture, mostly suitable for IoT applications. NodeMCU board uses ESP32 as processor and Wi-Fi module, has 32Mbit built-in flash and onboard PCB antenna. Peripheral interface options include UART, GPIO, ADC, DAC, SDIO, PWM, I2C, I2S. The power supply works with 5V, and the logic level is 3.3V. It uses Bluetooth 4.3 and works in the frequency range between 2400MHz-2483,5MHz. The pinout of the microcontroller is given in Figure 1.



3V3
PU EN
OD/ID S_VP ADC1_0 RTC GPIO36
OD/ID S_VN ADC1_3 RTC GPIO39
OD/ID VDET_1 ADC1_6 RTC GPIO34
OD/ID VDET_2 ADC1_7 RTC GPIO35
OD/ID 32K_XP TOUCH9 ADC1_4 RTC GPIO32
OD/ID 32K_XN TOUCH8 ADC1_5 RTC GPIO33
OD/ID DAC_1 ADC1_8 RTC GPIO25
OD/ID DAC_2 ADC2_9 RTC GPIO26
OD/ID TOUCH7 ADC2_7 RTC GPIO27
OD/IE/WPU HSPI_SCK MTMS TOUCH6 ADC2_6 RTC GPIO14
OD/IE/WPD VDD_FLASH HSPI_MISO MTDI TOUCH5 ADC2_5 RTC GPIO12
GND
OD/IE/WPD HSPI_MOSI MTCK ADC2_4 TOUCH4 RTC GPIO13
! OD/IE/WPU D2 GPIO9
! OD/IE/WPU D3 GPIO10
! OD/IE/WPU CMD GPIO11
5V0
EN
GND
GPIO23 VSPI_MOSI SPI_MOSI OD/IE
GPIO22 WIRE_SCL OD/IE
GPIO1 U0TXD SERIAL_TX OD/IE/WPU
GPIO3 U0RXD SERIAL_RX OD/IE/WPU
GPIO21 WIRE_SDA OD/IE
GND
GPIO19 VSPI_MISO OD/IE
GPIO18 VSPI_SCK OD/IE
GPIO5 VSPI_SS SDIO OD/IE/WPU
GPIO17 OD/IE
GPIO16 OD/IE
GPIO4 RTC ADC2_0 TOUCH0 OD/IE/WPD
GPIO0 RTC ADC2_1 TOUCH1 BOOT OD/IE/WPU
GPIO2 RTC ADC2_2 TOUCH2 OD/IE/WPD
GPIO15 RTC ADC2_3 TOUCH3 MTDO HSPI_SS LOG OD/IE/WPU
GPIO8 D1 OD/IE/WPU !
GPIO7 D0 OD/IE/WPU !
GPIO6 SCK OD/IE/WPU !
IO0
ESP-WROOM-32
USB Type-C
PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
DAC_X Digital-to-Analog Converter
DEBUG JTAG for Debugging
FLASH External Flash Memory (SPI)
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
ARDUINO Arduino Related Functions
STRAP Strapping Pin Functions
RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)
! Pin Shared with the Flash Memory Can't be used as regular GPIO
GPIO STATE
WPU: Weak Pull-up (Internal)
WPD: Weak Pull-down (Internal)
PU: Pull-up (External)
IE: Input Enable (After Reset)
ID: Input Disabled (After Reset)
OE: Output Enable (After Reset)
OD: Output Disabled (After Reset)

Figure 1. [2]

MPU-6050

The MPU6050 is an IMU (Inertial Estimation Unit) sensor. IMU sensors are used in a wide variety of applications such as mobile phones, tablets, satellites, spacecraft, drones, UAVs, robotics, and many more. They are used for motion tracking, orientation and position detection, flight control, etc.

The MPU6050 is a Micro-Electro-Mechanical Systems (MEMS) that consists of a 3-axis Accelerometer and 3-axis Gyroscope inside it meaning that this helps us to measure acceleration, velocity, orientation, displacement and many other motion-related parameters of a system or object. This module also has a (DMP) Digital Motion Processor inside it which is powerful enough to perform complex calculations and thus free up the work for Microcontroller.

The working principle is based on the Coriolis Effect. The Coriolis Effect states that when a mass moves in a particular direction with velocity and an external angular motion is applied to it, a force is generated and that causes a perpendicular displacement of the mass. The force that is generated is called the Coriolis Force and this phenomenon is known as the Coriolis Effect. [3]. The rate of displacement will be directly related to the angular motion applied. The Coriolis effect causes a vibration when the gyros are rotated about any of the axes. These vibrations are picked up by the capacitor. The signal produced is then amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is then digitized using ADC's. The DMP present on MPU6050 offloads the computation of motion-sensing algorithms from the host processor. DMP acquires data from all the sensors and stores the computed values in its data registers or in FIFO. FIFO can be accessed through the serial interface. Using AD0 pin more than one MPU6050 module can be interfaced with a microprocessor. MPU6050 can be used easily with Arduino, as MPU6050 has well-documented libraries available.[4] The pinout of the MPU6050 is given in Figure 2.

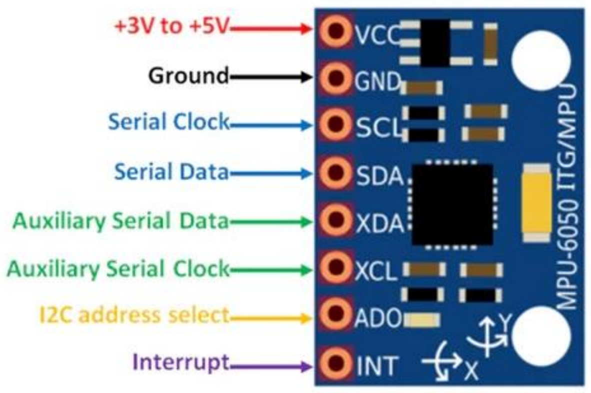


Figure 2. [5]

SG 90 Servo Motor

The SG90 Servo Motor is an actuator that produces a rotary motşon approximately from 0˚ to 180˚. Servo motors in general are used in many applications such as robotic arms, steering wheels, camera gimbals, etc. Servo motors have three wires: power, ground, and signal. The power wire is red, should be connected to the power supply, usually 5V. The ground wire is usually black, or brown should be connected to the ground. The signal wire is usually yellow, or orange should be connected to a PWM (Pulse Width Modulation) on the board.

Figure 3. [6]

The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Width Modulation. The length of the pulse will determine how far the motor turns. The microcontroller sends out PWM signals to the servo, and then the embedded board on the servo receives the signals through the signal pin and the controls the motor inside to turn. As a result, the motor drives the gear system and then rotates the shaft after deceleration. The shaft and the potentiometer of the servo are connected. When the shaft rotates, it drives the potentiometer, so the potentiometer outputs a signal to the embedded board. Then the board determines the direction and the speed of the rotation based on the current position, so it can stop exactly at the right position as defined and hold there.

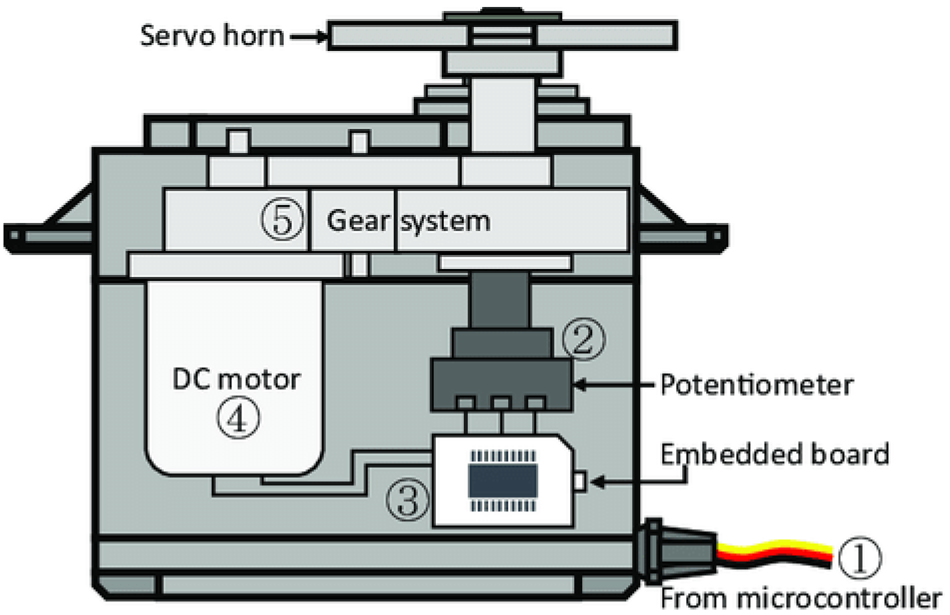


Figure 4. [6]

Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the left. ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, " the left.

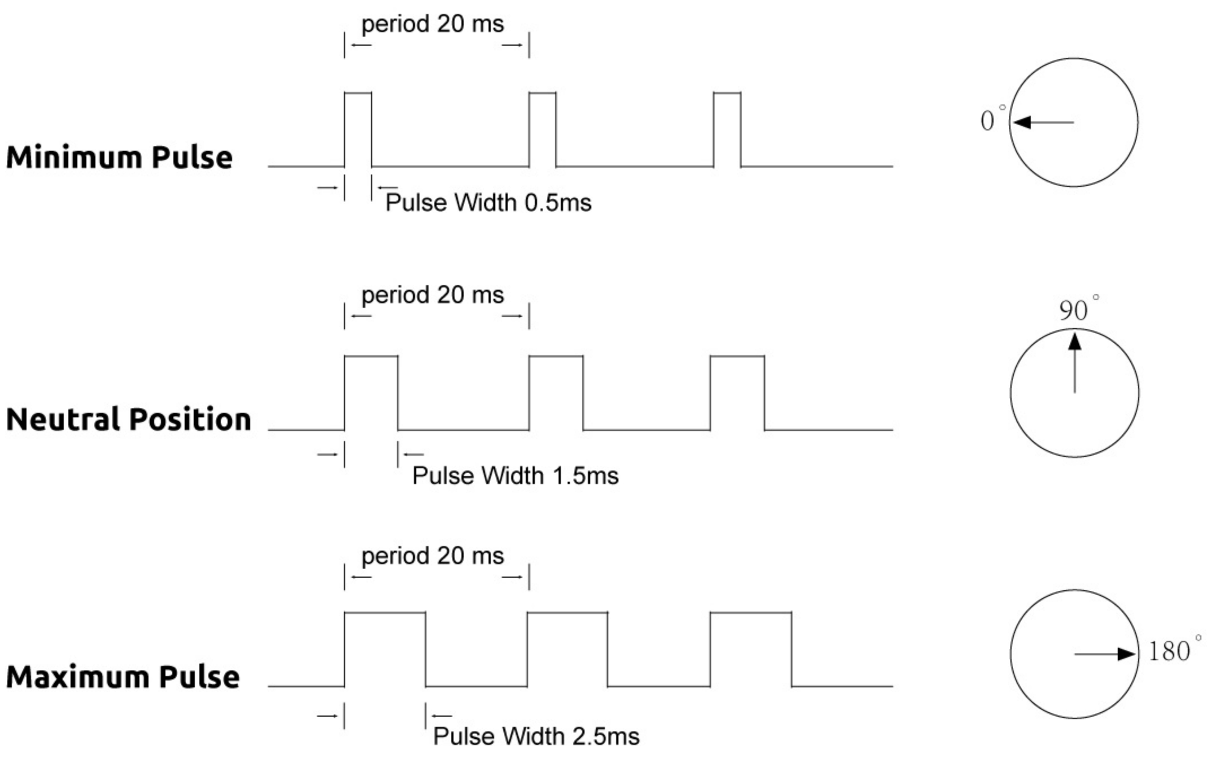


Figure 5. [6]

DRV8833

To control the speed and direction of the DC motor, we are going to use DRV-8833. The DRV8833 features two NMOS H-bridge drivers, enabling it to control two DC brush motors, a bipolar stepper motor, solenoids, and other inductive loads. The driver supply voltage is 2.7 to 10.8V and the allowable continuous current is 1.5A and the maximum allowable current is 2A. Some common applications are: POS printers, toys, robotics, office automation systems.

To change the direction of rotation of a DC motor you need to change the polarity of the supply. A common technique to do that is by using an H-bridge motor driver. An H-bridge motor driver consists of four switches (usually MOSFETs) arranged in a certain formation and the motor is connected to the center of the arrangement to form the H-like structure, by closing/enabling two opposite switches we can change the direction of the flowing current thus changing the direction of rotation.

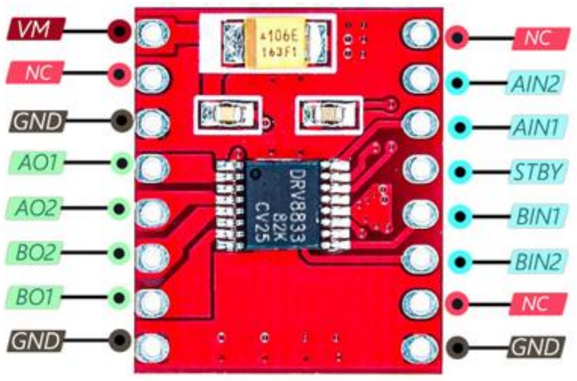


Figure 6. [7]

The pinout of the DRV-8833 is given in Figure 6. This module has 16 pins:

- VM: Motor voltage
- GND: Ground – three grounds connected to each other
- AO1: Positive end for motor A
- AO2: Negative end for motor A
- BO1: Positive end for motor B
- BO2: Negative end for motor B
- AIN1: Control signal for motor A

- AIN2: Control signal for motor A
- BIN1: Control signal for motor B
- BIN2: Control signal for motor B
- STBY: To active Standby mode, it should be HIGH.
- NC: Not used

Different modes of the control signals provide different performance modes for DC motors. Different modes are given in Table 1.

	AIN1 / (BIN1)	AIN2 / (BIN2)
Forward Direction	HIGH	LOW
Reverse Direction	LOW	HIGH
Brake / Stopped	LOW	LOW
Brake / Stopped	HIGH	HIGH

Table 1.

N20 DC Motor

N20 Micro Gear Motor is a versatile and compact DC gear motor that is used in various applications such as: robotics, camera gimbal, smart home devices, medical equipment and electronic devices. The N20 micro gear motor is designed to provide high torque output, making it suitable for our project that requires precise and powerful motion control.

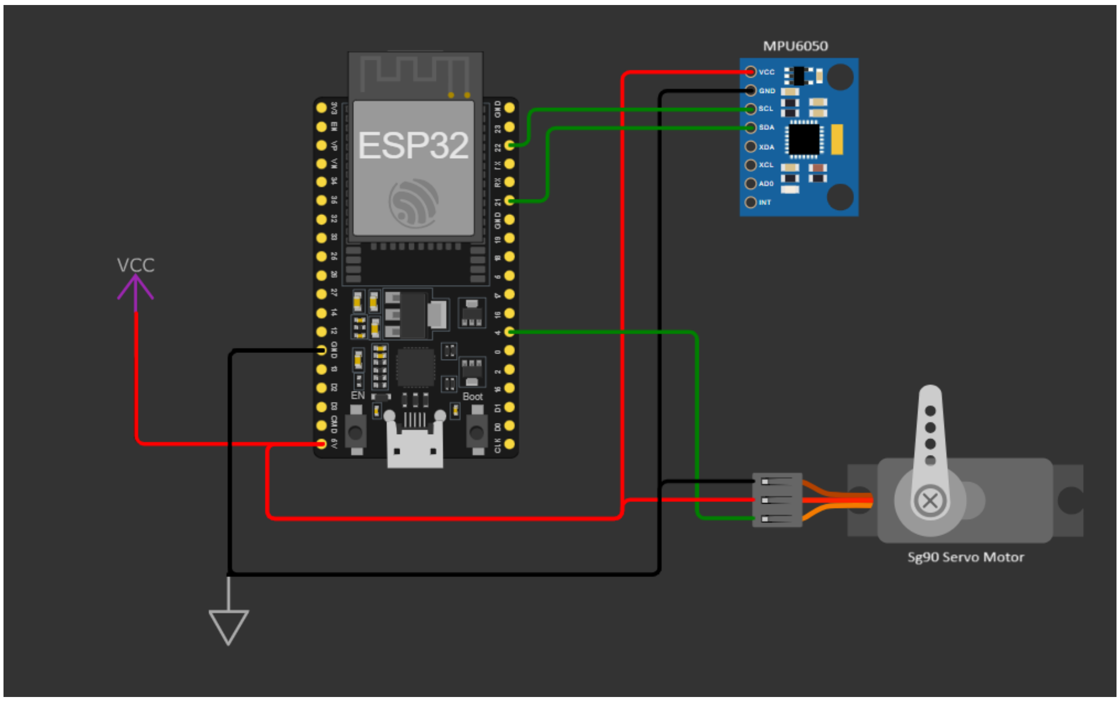
The N20 micro gear motor consists of several components, including the motor, gearbox, and output shaft. The motor is usually a small DC motor, while the gearbox is made up of several gears that are designed to work together to achieve the desired output speed and torque. The output shaft is the part of the motor that connects to the load and provides the motion control. The motor is powered by direct current, which is supplied by a battery or power supply. When the motor is powered, it rotates the gearbox, which in turn rotates the output shaft.[8] The gear ratio of the gearbox determines the speed and torque output of the motor. The higher the gear ratio, the slower the output speed and the higher the output torque.

Figure 7. [8]

Robot Construction (Circuitry)

There are 2 main parts of this robot. Direction control with servo motor and speed control with DC motor.

In the direction control part, we used MPU6050 Gyro and Acceleration Sensor and Sg90 Servo Motor.



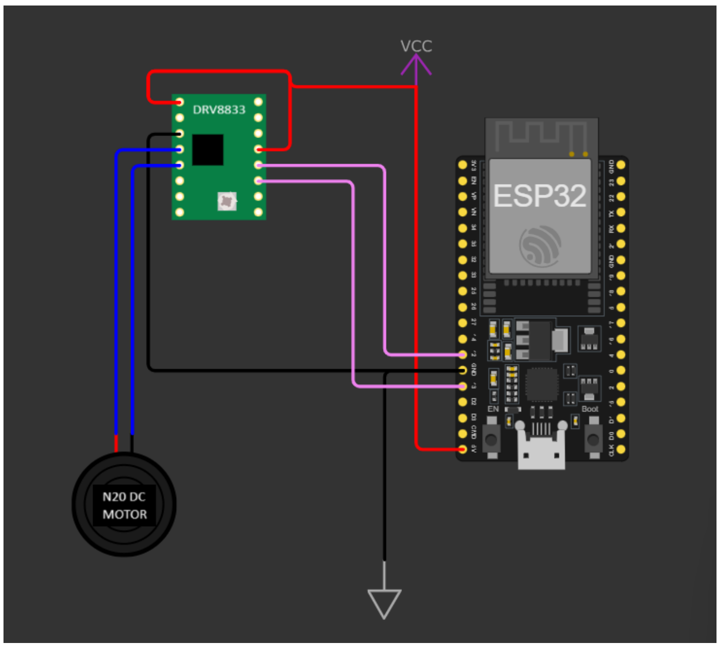
Schema 2

As can be seen from the Schema 2, we connected the MPU6050 to the 21 and 22 pins of the ESP32. These are SCL and SDA pins. SCL is for Serial Clock and SDA is for Serial Data.

SCL is used to synchronously clock data in or out of the target device. SDA used to transmit data to or from the target device. With these pins we will be able to take data according to the inclination of the bicycle. To find the inclination we used the angle between the y and z axis. Also, we take the angular velocity along the x axis by using the gyro value of that axis. Then we connected VCC pin to the 5V pin of the ESP32 and connected the GND.

There are 3 pins of the Servo Motor. We took the PWM value from the pin 4 of the ESP32. We took the inclination values from the MPU6050 and by using the correct PID implementation we produce a PWM value for servo motor. We controlled the servo motor movements by changing PWM. Again, we did the VCC and GND connections.

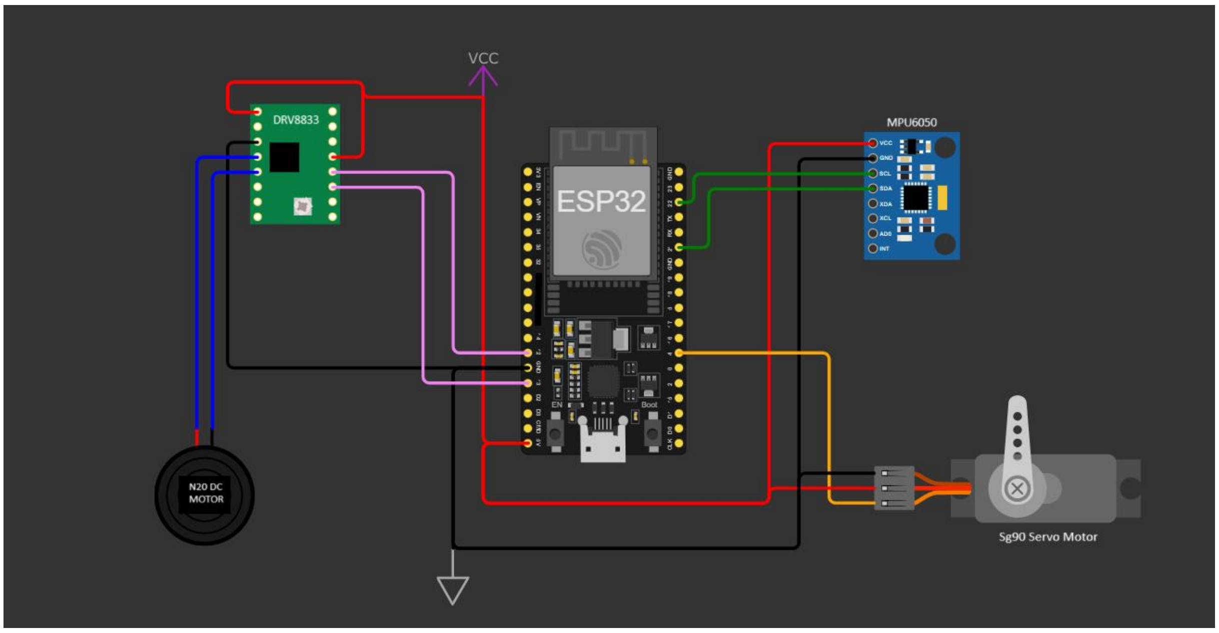
In the speed control part, we used N20 DC Motor and DRV3388 motor driver board.



Schema 3

We connected the AIN1 and AIN2 pins to the pin 12 and pin 13 of the ESP32. These pins to provide the speed of the motor. We gave 0 to one pin and desired speed amount (between 0 to 255) of PWM to the other pin. Which pin we send the PWM value determines the direction of the motor. We took the output from the AO1 and AO2 pins of the driver for the N20 motor. After that, we did the VCC and GND connections. However, there were some problems with the design of the bicycle. Thus, we could not be able to add this circuit to the robot. Which will be explained in the Results & Modifications part.

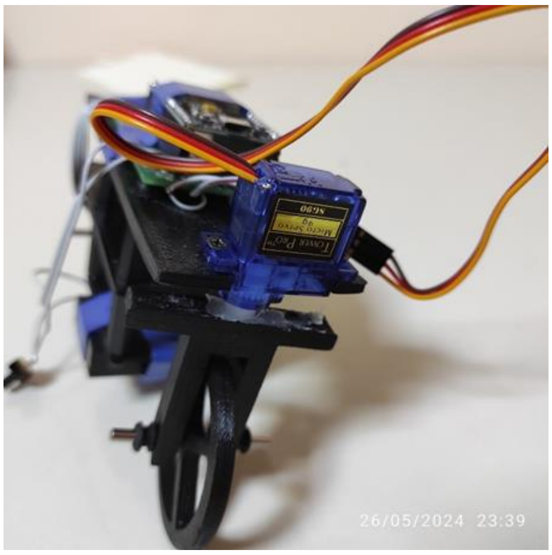
Since ESP32 has already provide bluetooth control, we did not need to use HC05. The final design is given in Schema 4 below.



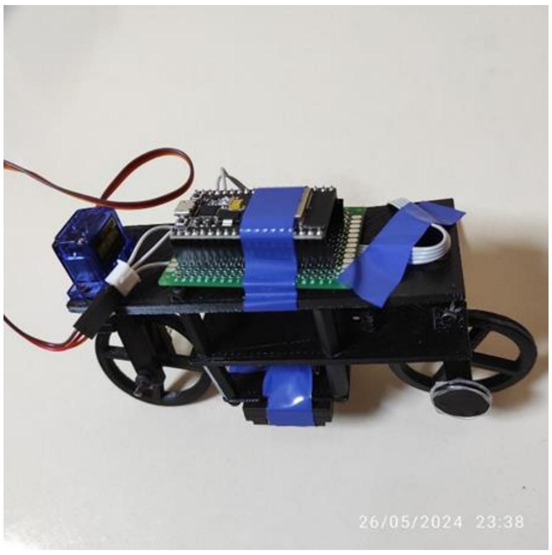
Schema 4

Mechanical Design

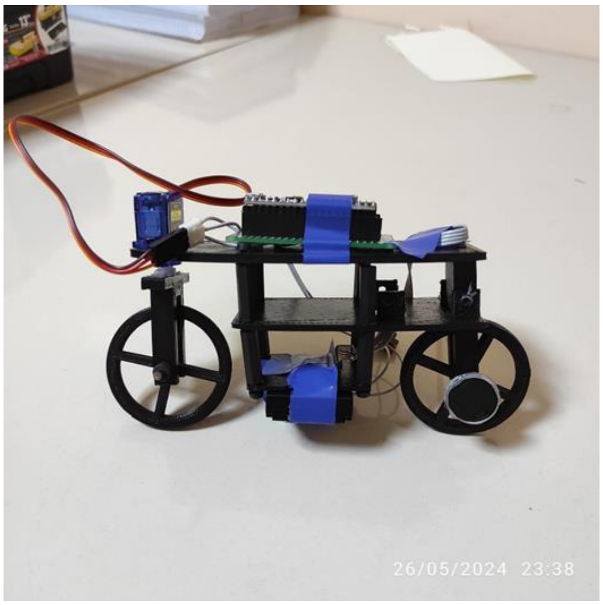
At first, thanks to our instructor to design the model of the bicycle and print it for us. Also, the solder of the circuit. We connected the servo motor directly to the forward wheel to provide the motion of the steering wheel. In order to take precise data from the MPU6050 we put the sensor bottom of the bicycle and just in the middle of the wheels. We needed constant current values for our robot to work properly. For that reason, we could not use a normal battery. Also, the Li-ion battery that we used could not make the circuit work. Thus, we put the ESP32 on the top of the bicycle, to connect it easily to the computer.



(a)



(b)



(c)

Figure 8.

Software

In our project we use PID control for balance. If the bicycle starts to fall over, the steering control tilts the front Wheel in that direction and small oscillations to maintain balance. For this system to work we need quick response time and sensitive readings of angle.

PID Control

PID control is a widely used control algorithm in industrial and engineering applications. It helps to maintain a desired setpoint by continuously calculating the difference between setpoint and measured value and applying a correction based on proportional, integral, and derivative terms. We can formulize it like this:

$$Output = Kp.e(t) + Ki \int e(t).dt + Kd.\frac{d}{dt}e(t)$$

$$\text{where } e(t) = setpoint - input$$

$$\begin{aligned} Output &= Kp * (error) + Ki * (errorSum) * dt + Kd * (currentError - previousError)/dt \\ &= Kp * (error) + Ki * (errorSum) * sampleTime - Kd * (currentAngle - previousAngle)/sampleTime \end{aligned}$$

Figure 9. [9]

To achieve quick response time, we set the Kp, Ki, Kd variables and sample time according to our goal.

Kp is proportional part of pid. It is current error and affected greatly from momentary changes. We made this value the highest because it provides instant change and increase oscillations.

Ki is integrational part of pid and It is concerned with the accumulation of past errors. It must be lower than kp but not too low because higher the kp / ki is faster the reponse time. But if ki is too small system has hard time approaching the set value.

Kd is derivate part of pid and it prevents overshooting by slowing the system when measured value get close to setpoint. We want to make this low as possible for fast response time and ensuring oscillations.

Sample time is the loop time of the code. We find this by making a last_time variable and everytime it loops back, writing serial the current time – last_time. After that we change the value of last_time to current time . By doing this we accurately find the sample time.

Calculating angle

For Calculating angle we use Complementary Filter that consist of high pass filter and low pass filter according to formula given below:

$$\boxed{\texttt{currentAngle} = \alpha.\underbrace{(\texttt{previousAngle} + \texttt{gyroAngle})}_{\text{HPF}} + \underbrace{(1-\alpha).(\texttt{accAngle})}_{\text{LPF}}} \qquad \boxed{\alpha = \frac{\tau}{\tau + dt}}$$

Figure 10. [9]

We give T = 0.004 because the less T value results in more sensitive angle reading. The value of dt is sample time.

Main Code review

In the code written for the bicycle balance, we use an ESP32 microcontroller to create a system that controls the balance with an MPU6050 sensor. First, the necessary libraries and variables are defined. The servo motor, MPU6050 sensor, variables required for PID (proportional integral-derivative) control and target angle of the controller are specified. In the configuration function, the serial and I2C communication are started, the servo motor is connected to the GPIO connector, and the MPU6050 sensor is initialized. In the loop function, the angle is calculated by reading the acceleration and gyro data received from the sensor. The angle obtained from the acceleration data and the angle obtained from the gyroscope data are combined to calculate the current angle. The errors are calculated by the PID control algorithm and the corresponding signal is sent to the servo motor. This signal causes the servomotor to move at a certain angle, which stabilizes the system. At each cycle, the motor power and servo motor movement are calculated and these values are printed on the serial monitor.

DC motor control

For controling DC motor, we use bluetooth to remotely controling via smartphone. First we use the Serial Bluetooth Terminal app to connect to Esp32. After that we use the terminal for control. But it is inefficient to use because every time we have to write to terminal to change somthing. According to that we changed the app we use to Dabble. Dabble offers speed control and interface with the buttons.

Control with Serial Bluetooth Terminal Code review

It allows us to control an LED via Bluetooth with the ESP32 microcontroller so that the bike can be controlled remotely. First, we included the BluetoothSerial library and created a BluetoothSerial object to manage the Bluetooth connection. We defined a character variable cmd to hold incoming commands. In the setup function, we initialized the Bluetooth connection with the name "Esp32-BT". We also set the GPIO 2 pin as output and used digitalWrite(2, HIGH) to make the LED connected to this pin light up at startup. In the loop function, we continuously checked the incoming data over Bluetooth. serialBT. available() function to check if there is data coming from the Bluetooth serial connection. If there is data, we read it with the serialBT.read() function and assign it to the cmd variable. Then, according to the value of the cmd variable, we set the state of the LED connected to the GPIO 2 pin. If cmd contains the character '1', we turn on the LED with the digitalWrite(2, HIGH) command. If cmd contains the character '0', we turned off the LED with the digitalWrite(2, LOW) command. As a result, with this code we were able to turn on and off an LED connected to the ESP32 with commands sent over Bluetooth. Thanks to this simple wireless control mechanism, we were able to control the state of the LED with '1' and '0' commands sent over Bluetooth.

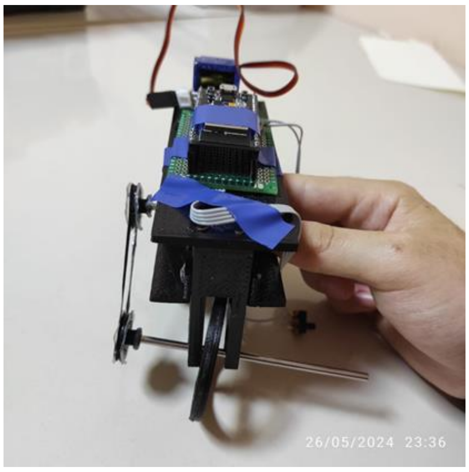
Controlling the Motor with Dabble code review

The code we wrote for the motor allows us to adjust the speed and direction of a motor using the ESP32 microcontroller and the Dabble mobile app. Initially, we include the necessary libraries and define the pins used for motor control. We used pin 25 for PWM and pins 26 and 27 to determine the motor directions. In the setup function, we start the serial communication and connect the ESP32 via Bluetooth using the Dabble library with the name "MyEsp32". In the loop function, we regularly update the data coming from the Dabble app. We process this data with the Dabble.processInput() function. We control the speed and direction of the motor according to the data we receive from the Dabble app. Controls.runMotor1() function sets the speed and direction of the motor. We control the motor using the L293D integration so that the motor runs according to commands from the mobile app. In conclusion, with this code we can use ESP32 and L293D integration to control the speed and direction of the motor through the Dabble app.

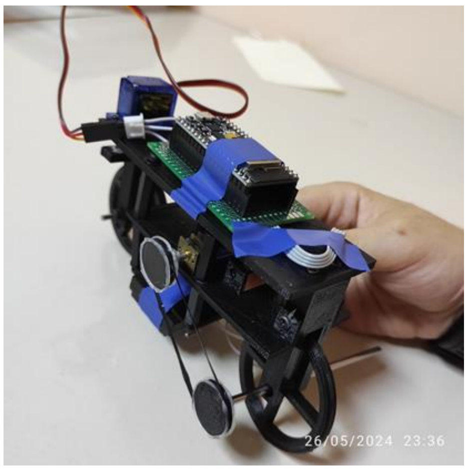
RESULTS & MODIFICATIONS

Strap Implementation

We could not be able to implement the strap on the bicycle, since the hole on the body for the back wheel was quite big. We were planning to connect the N20 DC motor and the back wheel with a strap. However, when we put the strap, the back wheel moved through the side where we put the N20 motor. This situation ruined the balance of the bicycle by changing the position of the back wheel. Since the position of the back wheel is ruined, when we started to work the N20 motor, the strap jump. Due to that reason, we could not add the speed control part of the circuit which is also given in Schema 3.



(a)



(b)

Figure 11.

Power Supply

There are three ways to supply power to ESP32: 5V pin, 3.3V pin and the Micro USB port on the development port. Since we are designing a bicycle, initially, we based our designs as the 5V pin connected to a 5V battery. Initially we have assumed the voltage of the batter was low and experimented with a higher voltage battery, but we have observed the same results. After measuring current and voltage values we have discover that the current from the battery was not stable enough to supply every component in our design. After experimenting while using Micro USB port, we have received a 3.3V battery, a step-up circuit, and a regulator circuit from our instructor. This way 3.3V would be converted to 5V and 5V would be constantly regulated. After connecting the components, we have observed that the battery had a short life, and it would only power up the circuity for about 10 seconds. Since the Micro USB had a stable current and voltage, we have continued to experiment with USB cable connected to the ESP32.

Wheels

The wheels of the bicycle had insufficient friction. Because of that we could not mount tires as we planned and could not increase the friction. This resulted in a vehicle that was more difficult to balance and we try to make up to it with adjusting the variables in the code to make balancing system more sensitive and making the circuit more symmetrical.

MPU6050 Offset

The library of mpu6050 is different for Ardunio and Esp32. Their content is different and they do not contain the same functions. Offset functions are among these functions so we could not make the offset calibrations.

CONCLUSION

The aim of this project was to build a bicycle robot that is remotely controlled and able to follow a straight line and being balanced by maneuvering the handlebar. To achieve a state of balance a gyro and acceleration sensor is used. From this sensor the inclination value of the bicycle is taken, and by using the PID implementation a PWM value is produced which is for the servo motor. Servo motor, turn the steering wheel through the inclined side. With the software, needed data, such as angle between the z and y axis and inclination amount through the x axis are taken from the gyro and acceleration sensor. 2 measurement result of angle are combined by using HPF and LPF. Error function is defined and the PID equation is written according to this. By changing the coefficients of this equation, the balance sensitivity is arranged. For some reasons which are already explained the speed control part could not be added. The mechanical design of the robot was made in order to help balance. The construction difficulties are observed and explained.

REFERENCES

- “The Most common types of Remote Control, infrared remotes, wifi remote, wired remote, Rf remotes”, *lripl.com, 2020.* https://www.lripl.com/blogdetails/the-most-common-types-of-remote-control#:~:text=The%20Most%20common%20types%20of,remote%2Cwired%20remote%2CRf%20remotes
- “Remote control (disambiguation)”, *en.wikipedia.org,* 2024. https://en.wikipedia.org/wiki/Remote_control_(disambiguation)
- “Remote control”, *en.wikipedia.com*, 2024. https://en.wikipedia.org/wiki/Remote_control
- “Diy self balancing robot arduino based.”, *youtube.com*, 2020. https://www.youtube.com/watch?v=aUbBUd-hBLI
- “ESP32 Web Server with MPU-6050 Accelerometer and Gyroscope (3D object representation)”, *youtube.com*, 2021. https://www.youtube.com/watch?v=dXcF-Uqa-gw
- “The PID Controller”, *wired.chillibasket.com*, 2015. https://wired.chillibasket.com/2015/03/pid-controller/

- Hobby, SriTu. “Controlling Servo Motor with Accelerometer | MPU6050 with Servo.” *SriTu Hobby*, 26 May 2021, https://srituhobby.com/controlling-servo-motor-with-accelerometer/

[1]
“ESP32 Wi-Fi & Bluetooth Modules I Espressif,” *Espressif.com*, 2024. https://www.espressif.com/en/products/modules/esp32 (accessed May 26, 2024).

[2]
“NodeMCU-32S, ESP32 WiFi+Bluetooth Development Board,” *Waveshare.com*, 2024. https://www.waveshare.com/nodemcu-32s.htm (accessed May 26, 2024).
[3]
“How Does the MPU6050 Accelerometer & Gyroscope Sensor Work and Interfacing It With Arduino,” *Circuitdigest.com*, 2023. https://circuitdigest.com/microcontroller-projects/interfacing-mpu6050-module-with-arduino (accessed May 26, 2024).

[4]

“MPU6050 : Pin Diagram, Circuit Working, Specifications & Applications,” *ElProCus - Electronic Projects for Engineering Students*, Sep. 05, 2019. https://www.elprocus.com/mpu6050-pin-diagram-circuit-and-applications/

[5]

“MPU-6050 Triple-Axis Accelerometer & Gyroscope Module,” *Robocraze*, 2022. https://robocraze.com/products/mpu-6050-triple-axis-accelerometer-gyroscope-module#:~:text=It%20works%20on%20the%20principle,try%20to%20tilt%20this%20arrangement. (accessed May 26, 2024).

[6]
“Servo Motor (SG90) — SunFounder Ultimate Sensor Kit documentation,” *Sunfounder.com*, 2024. https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/27-component_servo.html (accessed May 26, 2024).

[7]
“Interfacing DRV8833 Dual Motor Driver Module with Arduino - Electropeak,” *Electropeak*, Feb. 15, 2021. https://electropeak.com/learn/interfacing-drv8833-dual-motor-driver-module-with-arduino/ (accessed May 26, 2024).
[8]
“N20 Micro Gear Motor: The Powerhouse of Robotics and DIY Projects,” *Robomart.com*, 2023. https://robomart.com/blog/n20-micro-gear-motor-the-powerhouse-of-robotics-and-diy-projects#:~:text=The%20mechanism%20of%20an%20N20,the%20higher%20the%20output%20torque. (accessed May 26, 2024).
[9]

“Arduino Self-Balancing Robot” *instructables.com*, 2017.
https://www.instructables.com/Arduino-Self-Balancing-Robot-1/ (accessed May 26, 2024).