

Microcontroller Design Direction

[Christian Hansis](mailto:christian.hansis@njit.edu)

Students:

Arturo Garcia

Joao Sarmiento

Jeury Abreu

2024-09-31

# **Microcontrollers:**

## Arduino Uno:

* Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.

## S32:

* The S32 chip refers to a family of microcontrollers and processors developed by NXP Semiconductors, designed specifically for automotive, industrial, and safety-critical applications. The S32 family is known for its high-performance, real-time capabilities and its support for automotive-grade standards such as ISO 26262.

## ESP 32:

* ESP32 is a chip that provides Wi-Fi and (in some models) Bluetooth connectivity for embedded devices – in other words, for IoT devices. While ESP32 is technically just the chip, the modules and development boards that contain this chip are often also referred to as “ESP32” by the manufacturer.

# Pros and Cons:

| Arduino Uno (Atmega328P): | |
| --- | --- |
| Pro | Con |
| Beginner Friendly | Limited Processing Power (Single-Core 16MHz) |
| Versatile and Well-Supported | Limited Memory(2KB SRAM, 32KB Flash, 1KB EEPROM) |
| Affordable | No Built-in Wi-Fi or Bluetooth |
| Extensive Shield Ecosystem | Limited I/O Pins |
| Open-Source Hardware and Software | No Real-Time Operating System (RTOS) |
| Low Power Consumption | Not Ideal for Resource Heavy Projects |
| USB Powered |  |

## 

| S32: | |
| --- | --- |
| Pro | Con |
| Automotive-Grade Quality | High Complexity |
| High Performance | Higher Cost |
| Scalability | Overkill for Non-Automotive Applications |
| Real-Time Processing | Development Complexity |
| Advanced Security Features | Longer Development Cycle |
| Power Efficiency | Power Consumption in High-Performance Modes |
| Wide Ecosystem Support | Limited Availability in Consumer Markets |

## 

| ESP32: | |
| --- | --- |
| Pro | Con |
| Wi-Fi and Bluetooth Connectivity | Complexity |
| Dual Core Processor | Limited Pin Availability |
| Low-Cost | Limited Ram (520 KB SRAM) |
| GPIO and Peripheral Support | Overheating Under Heavy Load |
| Low Power Consumption modes | High Power Consumption in Active Mode |
| Open Source SDK | Steeper Learning Curve |
| Built-in Security Features | Software Complexity |

# **Decision:**

* Our decision is to use the Arduino Uno for our Mono-wheel as it is a good choice for simple, small-scale projects, especially for beginners due to its affordability, ease of use, and extensive community support. It’s ideal for basic tasks, low-power applications, and projects that don't require significant processing power, memory, or multitasking.

# 

# **Use of Resources from Microcontroller:**

### **1. Motor Control (PWM via Timer):**

**Resource:** Digital I/O Pins (PWM Capable), Timer1

#### **PWM Parameters for Mono-Wheel Motor:**

* **Duty Cycle**: The duty cycle controls the speed of the mono-wheel motor by adjusting the average voltage supplied to the motor. The duty cycle is a percentage of time the signal stays HIGH (on) versus LOW (off).
  + **Initial Duty Cycle**: Start with 50% (0.5) for testing.
  + **Adjustable Duty Cycle**: Can range between 0% (full stop) to 100% (full speed).
* **Frequency**: The frequency determines how fast the PWM signal switches between HIGH and LOW.
  + For motor control, a typical frequency is in the range of 500 Hz to 1 kHz, as lower frequencies can cause the motor to produce audible noise.
  + **Chosen Frequency**: 1 kHz for smoother motor control and less noise.

#### **Arduino Uno PWM Pin:**

* The use **pin D9**, which is one of the PWM-capable pins connected to **Timer1**. Timer1 is 16-bit, so it gives better control over the PWM signal compared to Timer0 and Timer2, which are 8-bit.

**2. Direction Control (Digital Output Pins):**

**Resource: Digital I/O Pins (GPIO)**

* The motor’s direction will be controlled by an H-Bridge driver ( L298N), which requires two digital pins to set the direction (forward or backward).

#### **Pin Assignment for Direction Control:**

* **Pin D7**: Forward direction
* **Pin D8**: Reverse direction

#### **Parameters:**

* The logic for these pins is binary:
  + **Forward**: D7 = HIGH, D8 = LOW
  + **Reverse**: D7 = LOW, D8 = HIGH

### **3. Sensor Feedback (Analog Input for Stability):**

**Resource: Analog I/O Pins**

* For the mono-wheel’s balance and stability, I will use an **IMU (Inertial Measurement Unit)** or accelerometer/gyroscope combination, connected to the analog inputs.

#### **Pin Assignment:**

* **Pin A0 and A1**: Used for analog input from the IMU (for X and Z axis).

#### **Parameters:**

* The IMU will provide feedback on the tilt and orientation of the mono-wheel. This data will be used to adjust the motor speed and direction to maintain balance.

### 

### **4. User Input (Potentiometer for Speed Control)**

**Resource: Analog Input or Digital Input**

* A **potentiometer** will allow the user to manually control the speed of the mono-wheel by adjusting the PWM duty cycle.

#### **Pin Assignment:**

* **Pin A3**: Analog input for the potentiometer (speed control).

#### **Parameters:**

* The potentiometer will adjust the motor speed by changing the duty cycle of the PWM signal.
* **Output Voltage Range**: 0-5V (corresponds to 0%-100% duty cycle).