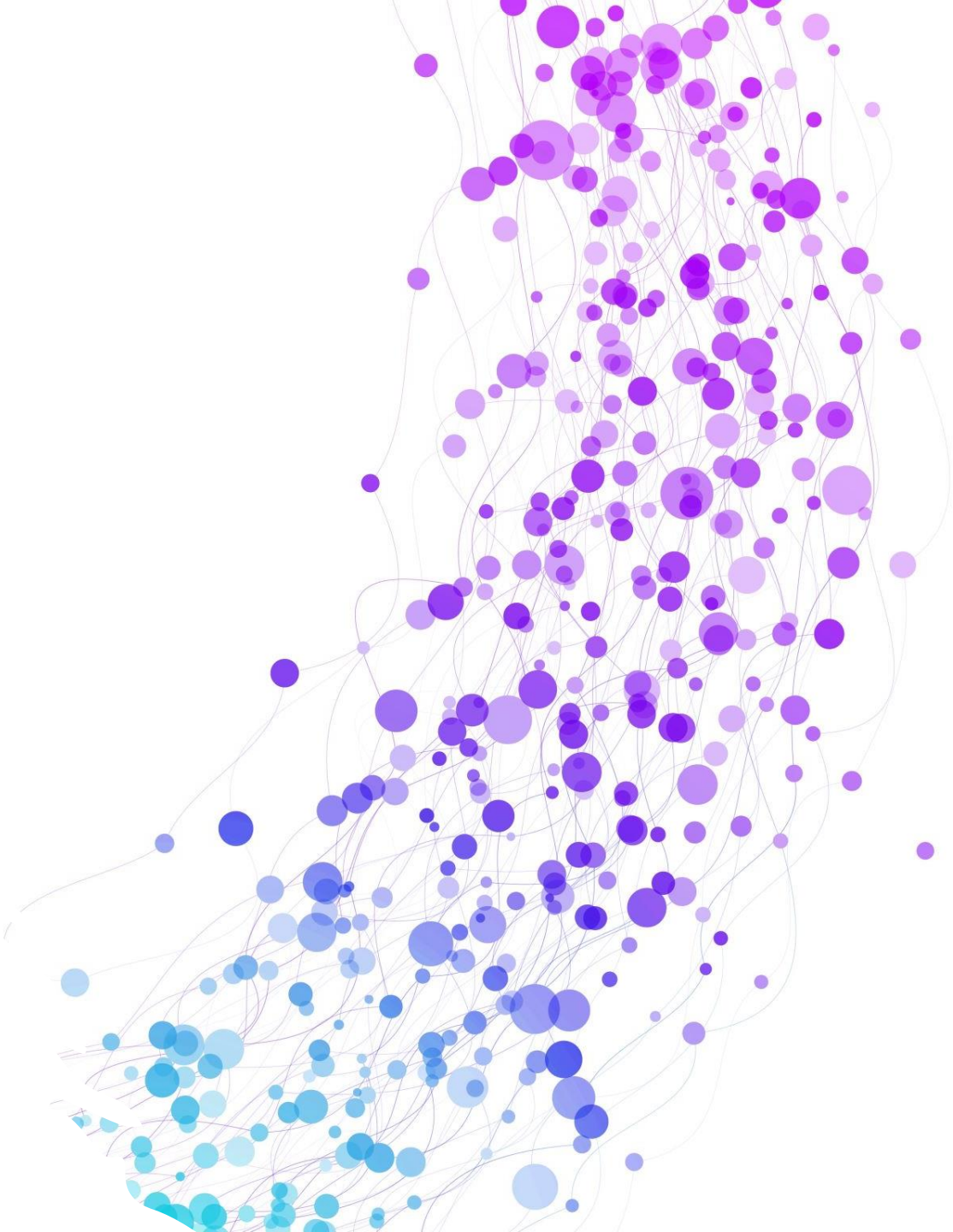
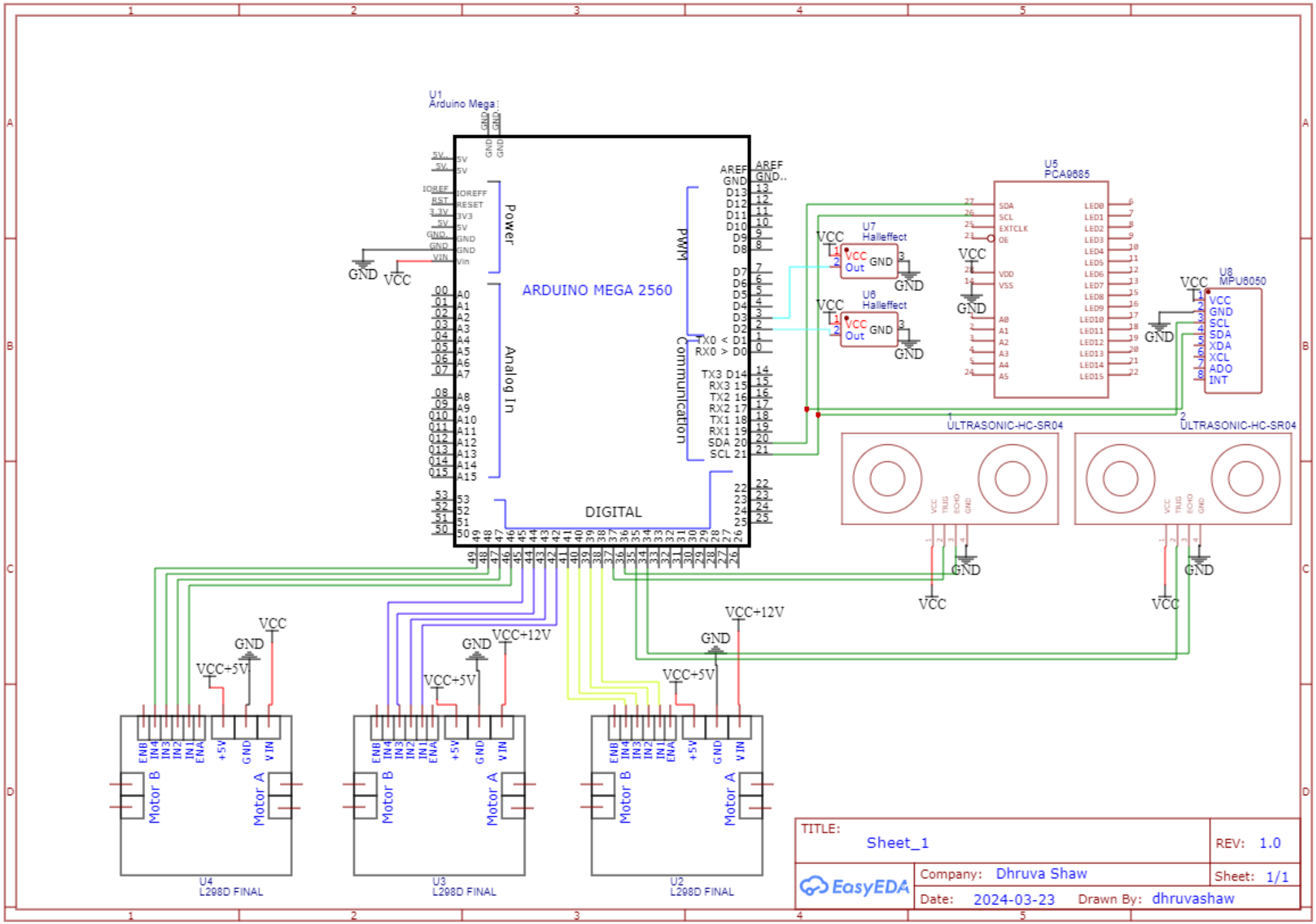
# India Skills 24 (State Track 2)

- Dhruva Shaw [LPU, Jalandhar]
- Abhishek [Nirvan Roopam Modern School]




U1
Arduino Mega:
ARDUINO MEGA 2560
Power
Analog In
DIGITAL
PWM
Communication
U5
PCA9685
U7
Halleffect
U6
Halleffect
U8
MPU6050
ULTRASONIC-HC-SR04
ULTRASONIC-HC-SR04
U4
L298D FINAL
U3
L298D FINAL
U2
L298D FINAL
Motor A
Motor B
VCC
GND
VCC+5V
VCC+12V
TITLE:
Sheet_1
REV: 1.0
Company: Dhruva Shaw
Sheet: 1/1
Date: 2024-03-23
Drawn By: dhruvashaw
EasyEDA

# Path Planning

Since the arena layout is predefined and static, we can leverage this knowledge to simplify the robot's control system. Path planning algorithms like Dijkstra's or A* are not essential in this scenario.

Because the environment and object locations are known beforehand, the robot's path can be pre-programmed, eliminating the need for real-time pathfinding.

This pre-programmed path can then be directly encoded into the robot's control program.

# Mobility Management

- Given the static and predetermined arena layout, we can optimize the robot's control system. Two primary control approaches are well-suited for this scenario: PID (Proportional-Integral-Derivative) and PI (Proportional-Integral) controllers. The optimal choice depends on the task at hand and time constraints.

- For time-sensitive tasks, PID control might be less favorable due to the initial tuning process required to find optimal constant values.  However, PI control offers a simpler and faster implementation.

- To simplify the control algorithm further, we can leverage the known environment.  Since we can track the robot's position through encoder ticks, resetting these ticks after each turn allows for a more manageable control strategy.

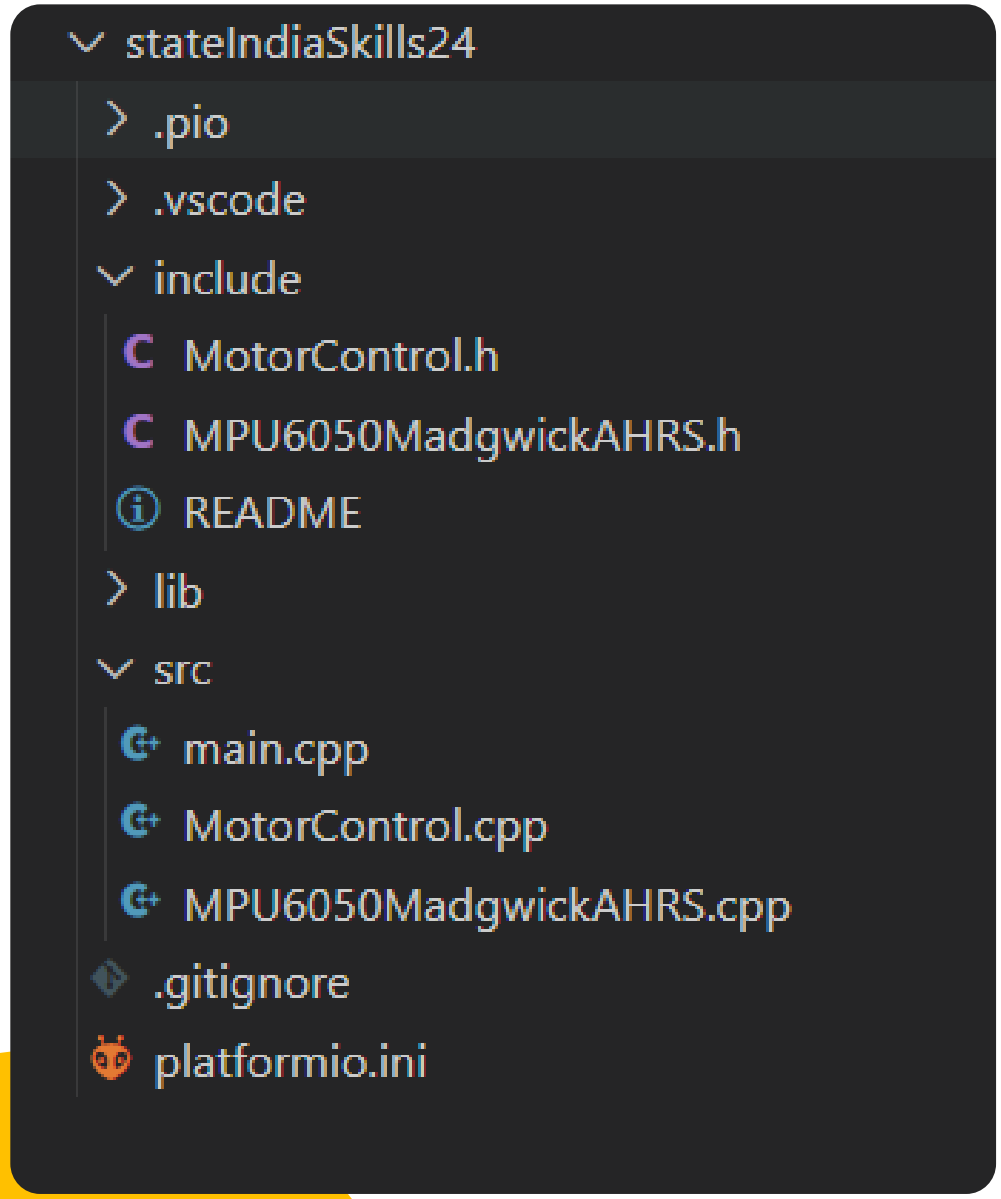
# Object Management

- Infrared (IR) Sensor: This sensor, positioned at the front of the chassis, will be activated upon reaching designated ball detection zones.  It will be primarily responsible for identifying balls within the environment.

- Ultrasonic Distance Sensor: This sensor plays a crucial role in obstacle avoidance. It not only detects the presence of walls but also provides an approximate distance measurement. This information allows the robot to maintain a central position within the designated path.

# Overall Task Analysis/Execution Strategy

- Leveraging a Known Arena for Efficient Control: Since the arena layout is predefined and static, we can exploit this knowledge to simplify the robot's control strategy. This approach streamlines the execution process and reduces complexity.
- Simplifying Control with Encoder Ticks: To further simplify the control algorithm, we utilize encoder ticks from the robot's two motors. Given the known environment, resetting these ticks after each turn allows for a more manageable control system. Since high precision isn't crucial in this scenario, using only two encoders is sufficient.
- Ultrasonic Sensor for Safe Navigation: For safe navigation within the arena, the robot employs an ultrasonic distance sensor. This sensor not only detects the presence of walls but also provides an approximate distance measurement. This real-time information allows the robot to maintain a central position within the designated path.
- Optional: Advanced Sensor Fusion (Depending on Task): While not essential for all tasks, the robot can be equipped with an MPU6050 sensor combining an accelerometer and gyroscope. When fused with the Madgwick's AHRS algorithm, this sensor can provide data on the robot's roll, pitch, and yaw. However, it's important to note that raw sensor data from this source can accumulate errors over time, and its usefulness depends on the specific task requirements.




# Code Analysis/Execution Strategy

- Modular Code Structure for Enhanced Maintainability
- To promote code clarity, maintainability, and reusability, we adopted a modular approach using Arduino's code abstraction strategy. This strategy involves:
- Function Separation: Individual functionalities are encapsulated within separate functions, enhancing code readability and reducing redundancy.
- Organized Files: These functions are further organized into dedicated files, promoting better code structure and easier navigation.
- API for Sensor Data: A dedicated Application Programming Interface (API) houses all the necessary functions for sensor data pre-processing and retrieval. This centralizes sensor data management and simplifies access in the main program.
- Main File Integration: The API is then seamlessly integrated into the main program file, ensuring essential functionalities are readily available when needed.
- PlatformIO Development Environment: PlatformIO, an extension for Visual Studio Code, provided the development environment for writing and managing the code. This streamlined the development process and offered additional functionalities.

# Thank You

We up for the Q/A session