STEM & Robotics Curriculum: 10 Hands-On Projects

A Structured Project-Based Learning Guide (Beginner → Expert)

---

1. Curriculum Overview

This curriculum is designed to introduce students to **STEM, electronics, robotics, IoT, and AI** through **10 progressively challenging projects**. Each project builds on skills learned previously and culminates in **assessment projects** that combine concepts into real-world applications.

- **Beginner Level (Projects 1–2):** Foundations of electronics, sensors, and basic robotics.
- **Intermediate Level (Projects 3–5):** IoT, wireless communication, automation, and mobile control.
- **Advanced Level (Projects 6–8):** Complex robotics, sorting automation, navigation, self-parking.
- **Expert Level (Projects 9–10):** Wearable robotics, AI vision, human-robot interaction, advanced autonomy.

Each level includes an **Assessment Project** to consolidate and test skills.

10 Core Main Projects

Beginner Level

Project 1: LED Traffic Light System

- Learn sequencing, timing logic, and basic circuits.
- Build a working traffic light with pedestrian button and optional buzzer.

Project 2: Line-Following Robot

- Learn sensor calibration, control logic, and robotics basics.
- Build a robot that follows a track using IR sensors.

Intermediate Level

Project 3: Smart Home Automation

- Learn IoT basics, sensors, and remote appliance control.
- Build a Wi-Fi-based home automation system with sensors.

Project 4: Obstacle-Avoiding Robot

- Learn ultrasonic sensing and servo-based scanning.
- Build a robot that avoids objects in real time.

Project 5: Bluetooth-Controlled Car

- Learn mobile integration and wireless control.
- Build a smartphone-controlled robotic car.

Advanced Level

Project 6: Robotic Sorting Machine

- Learn automation and conditional logic.
- Build a conveyor system that sorts items by color.

Project 7: Wi-Fi Rover

- Learn IoT robotics and live video streaming.
- Build a Wi-Fi-controlled rover with camera interface.

Project 8: Self-Parking Car

- Learn sensor fusion and motion planning.
- Build a robotic car that can parallel park autonomously.

Expert Level

Project 9: Robotic Exoskeleton Glove

- Learn biomechanics and wearable robotics.
- Build a glove that controls a robotic hand using flex sensors.

Project 10: AI Vision Rover (Capstone)

- Learn computer vision, Python, and advanced robotics.
- Build a Pi-based rover that uses AI for object detection and navigation.

Assessment Projects

Level Assessments (Integrated)

- **A1 (Beginner): Smart Pedestrian Crosswalk**
  A traffic light + line-following robot car system with pedestrian button.
- **A2 (Intermediate): IoT Obstacle-Avoiding Car**
  Bluetooth car with ultrasonic avoidance and Wi-Fi dashboard control.
- **A3 (Advanced): Intelligent Sorting & Delivery Rover**
  Wi-Fi rover with sorting conveyor and self-parking function.
- **A4 (Expert): Vision-Guided Rover with Glove Control**
  AI rover controlled by wearable glove, with switchable autonomous vision mode.

Individual Project Assessments (A5–A10)

- **A5 (for Project 1): Interactive Traffic City**
  Multi-intersection simulation with working lights and pedestrians.
- **A6 (for Project 2): Warehouse Delivery Robot**
  Line-follower that stops at delivery points.
- **A7 (for Project 3): IoT Weather Dashboard**
  Live temperature/light data to Wi-Fi dashboard with appliance automation.
- **A8 (for Project 4): Maze-Solver Robot**
  Ultrasonic robot navigating and solving a maze.
- **A9 (for Project 5): Smartphone Racing Challenge**
  App-controlled cars race on a custom track.
- **A10 (for Project 6): Recycling Station**
  Sorting machine simulating a recycling plant with LCD stats.

Learning Pathway

- **Beginner (Projects 1–2 + A1/A5/A6)**
  Foundations of electronics + intro to robotics.
- **Intermediate (Projects 3–5 + A2/A7–A9)**
  IoT, wireless communication, robotics integration.
- **Advanced (Projects 6–8 + A3/A10)**
  Real-world robotics, automation, navigation, system integration.
- **Expert (Projects 9–10 + A4)**

- AI, computer vision, wearable robotics, human-robot interaction.

Final Capstone Integration

At the **end of the curriculum**, students complete the **Capstone Challenge**:

Smart Robotic Ecosystem

- A Wi-Fi-enabled robotic rover with AI vision,
- Controlled via **robotic glove gestures**,
- Capable of **sorting items**, navigating obstacles, and self-parking,
- Integrated into a **smart environment** (traffic lights + home automation).

This **mega project** demonstrates mastery of **all skills across Beginner → Expert levels**.

Beginner Level

Project 1: LED Traffic Light System

- **Materials:** Arduino Uno, LEDs, resistors, breadboard, jumper wires.
- **Skills Learned:** Basic electronics, sequencing, timing logic.
- **Timeframe:** 1–2 weeks
- **Steps:** Wire LEDs, program traffic light sequence with delays, extend with pedestrian button.
- **Extensions:** Add buzzer for accessibility, simulate real-world traffic cycles.

Project 2: Line-Following Robot

- **Materials:** Arduino Uno, IR sensors, L293D motor driver, DC motors + chassis, battery.
- **Skills Learned:** Sensors, feedback loops, robotics basics.
- **Timeframe:** 2–3 weeks
- **Steps:** Mount sensors, write code to follow black line, test on track.
- **Extensions:** Add PID control for smoother tracking, race competition.

Intermediate Level

Project 3: Smart Home Automation

- **Materials:** Arduino Uno/ESP8266, relay module, DHT11 sensor, light sensor, Wi-Fi module.
- **Skills Learned:** IoT basics, automation, remote control.
- **Timeframe:** 3–4 weeks
- **Steps:** Control lights/appliances, add temperature/humidity readings, connect via Wi-Fi.
- **Extensions:** Add smartphone dashboard, integrate with Google Home/Alexa.

Project 4: Obstacle-Avoiding Robot

- **Materials:** Arduino Uno, ultrasonic sensor (HC-SR04), servo, motor driver, DC motors.
- **Skills Learned:** Real-time sensing, robotics control.
- **Timeframe:** 3–4 weeks
- **Steps:** Mount ultrasonic sensor on servo, scan for obstacles, move robot accordingly.
- **Extensions:** Add multiple sensors for 360° detection, integrate AI pathfinding.

Project 5: Bluetooth-Controlled Car

- **Materials:** Arduino Uno, HC-05 Bluetooth module, motor driver, DC motors + chassis, smartphone.
- **Skills Learned:** Wireless communication, mobile integration.
- **Timeframe:** 2–3 weeks
- **Steps:** Pair Arduino with phone, build control app, map commands to motion.
- **Extensions:** Add headlights/horn, integrate voice control.

Advanced Level

Project 6: Robotic Sorting Machine

- **Materials:** Arduino Mega, color sensor (TCS34725), servos, DC motor, conveyor system, frame.
- **Skills Learned:** Automation, conditional logic, mechanical design.
- **Timeframe:** 3–4 weeks
- **Steps:** Build conveyor, mount color sensor, program sorting logic, test with objects.
- **Extensions:** Sort by size/weight, add LCD counter for statistics.

Project 7: Wi-Fi Rover

- **Materials:** ESP32/ESP32-CAM, motor driver, camera module, DC motors, smartphone control interface.
- **Skills Learned:** IoT, wireless control, live video streaming.
- **Timeframe:** 3–4 weeks
- **Steps:** Assemble rover chassis, wire motors + ESP32, upload Wi-Fi control + camera stream code, build phone web interface.
- **Extensions:** Add obstacle avoidance, joystick control, telemetry dashboard.

Project 8: Self-Parking Car

- **Materials:** Arduino Mega, ultrasonic sensors, steering servo, DC motors + chassis.
- **Skills Learned:** Sensor fusion, motion planning, real-world robotics applications.
- **Timeframe:** 4–6 weeks
- **Steps:** Build car with steering, mount ultrasonic sensors, program parallel parking routine, test in mini parking lot.
- **Extensions:** Add multiple parking modes, integrate camera for vision-guided parking.

Expert Level

Project 9: Robotic Exoskeleton Glove

- **Materials:** Arduino Nano, flex sensors, servo motors, 3D-printed robotic hand, glove.
- **Skills Learned:** Wearable robotics, biomechanics, human-robot interaction.
- **Timeframe:** 5–6 weeks
- **Steps:** Attach flex sensors to glove, mount servos on hand, connect tendons, program glove-to-robot mapping.
- **Extensions:** Add wireless Bluetooth, haptic feedback, or IMU-based gesture recognition.

Project 10: AI Vision Rover (Capstone)

- **Materials:** Raspberry Pi 4, Pi camera, motor driver, robot chassis, OpenCV.
- **Skills Learned:** AI & computer vision, Python programming, advanced robotics integration.
- **Timeframe:** 6–8 weeks
- **Steps:** Assemble rover, set up Raspberry Pi with OpenCV, integrate motors + camera, train/test vision models.
- **Extensions:** Implement object following, traffic sign detection, or ROS for advanced AI control.

Contents

BEGINNER'S LEVEL

Project 1: LED Traffic Light System

Aim

To design and implement a miniature traffic light system using Arduino that simulates real-world traffic signaling, introducing students to basic electronics, programming logic, and automation concepts.

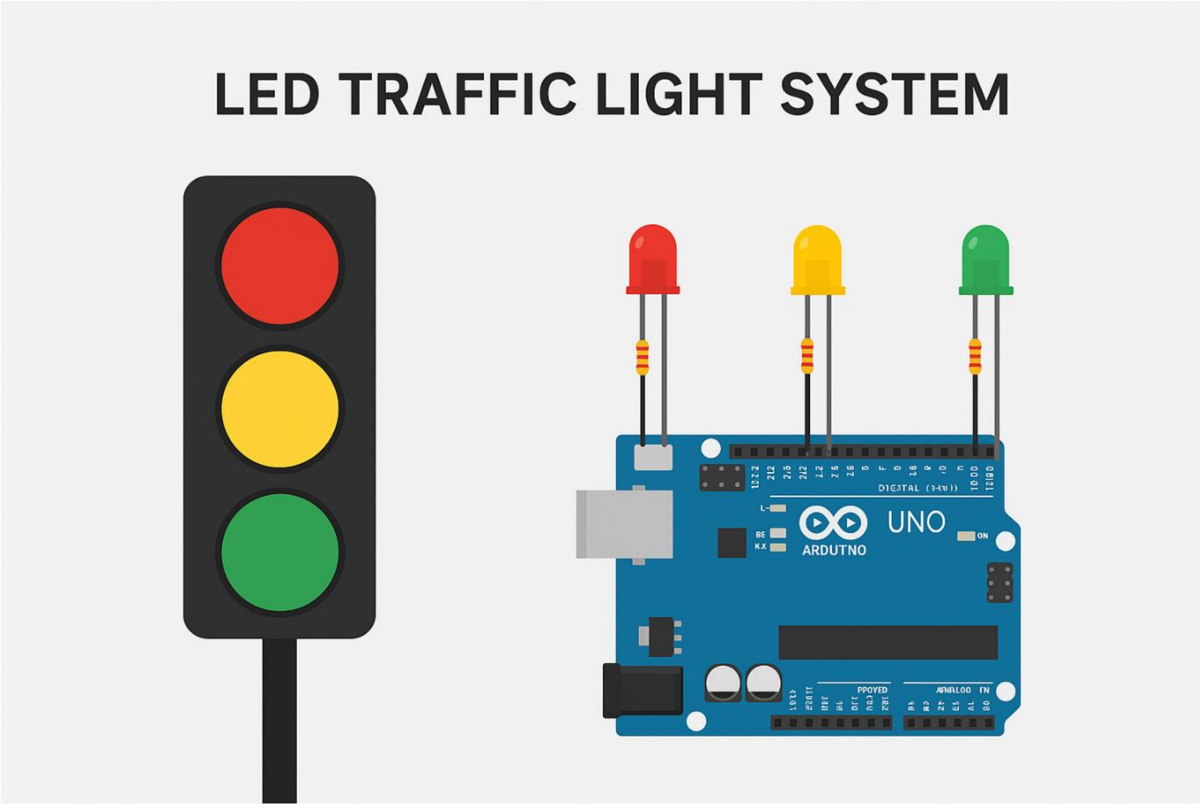
Objectives

1. Introduce students to Arduino microcontrollers and how they control electronic circuits.
2. Teach the fundamentals of **LED wiring, resistors, and breadboard prototyping**.
3. Demonstrate the use of **timing functions** (delays) to create sequences.
4. Highlight the importance of **traffic systems in real-world applications**.
5. Provide hands-on experience with coding simple logic for automated processes.

Goal

By the end of this project, students will:

- Understand how traffic lights function and why they are critical for road safety.
- Be able to **design, build, and program** a simple traffic light system with Arduino.
- Gain confidence in working with LEDs, breadboards, and simple control logic.
- Develop problem-solving skills by extending the project with real-world features (pedestrian button, buzzer).



Materials Required

- Arduino Uno (1x)
- Breadboard (1x)
- LEDs (3x: Red, Yellow, Green)
- Resistors (3x: 220Ω recommended)
- Jumper wires (male-to-male, ~10–15 pieces)
- USB cable (for uploading code)
- Optional: Buzzer + Push button (for extensions)

Skills Learned

- **Electronics:** LED polarity, resistor use, breadboard wiring.
- **Programming:** Sequencing with Arduino IDE (C/C++), use of `digitalWrite()` and `delay()`.
- **Logic Design:** Traffic cycle planning (Red → Green → Yellow).
- **Systems Thinking:** Applying technical skills to simulate real-world problems.

Timeframe

- **Week 1:** Introduction to Arduino + Breadboard basics, LED blinking exercise.
- **Week 2:** Building the full traffic light circuit + writing traffic light program.
- **Optional Extension Week:** Add pedestrian button + buzzer, simulate real-world scenarios.

Step-by-Step Process

1. **Circuit Assembly**
   - Place Red, Yellow, and Green LEDs on the breadboard.
   - Connect each LED through a resistor to digital pins on Arduino (e.g., Red → Pin 13, Yellow → Pin 12, Green → Pin 11).
   - Connect the ground of all LEDs to Arduino GND.
2. **Programming Traffic Sequence**
   - Open Arduino IDE.
   - Write code using `digitalWrite()` to turn LEDs on/off in order.
   - Insert `delay()` commands to simulate time gaps (e.g., Red 5s, Green 5s, Yellow 2s).
3. **Uploading & Testing**
   - Connect Arduino to PC with USB.
   - Upload the program.
   - Observe traffic light cycling automatically.
4. **Adding Pedestrian Button (Optional)**
   - Wire push button to a digital input.
   - Modify code to interrupt traffic cycle when button is pressed → trigger pedestrian crossing sequence.
5. **Adding Buzzer (Optional)**
   - Connect buzzer to a digital pin.
   - Program buzzer to beep during pedestrian crossing for accessibility.

Extensions

- Add **buzzer** to provide audio cues (simulating crosswalk beeps).
- Use **push button** to allow pedestrians to request crossing.
- Create **4-way traffic intersection** with multiple Arduino outputs.
- Replace delays with **millis() function** for better real-time control (advanced).
- Expand into a **mini smart city model** with multiple synchronized lights

---

Project 2: Line-Following Robot

Aim

To design and build an Arduino-based robot that can detect and follow a black line on a white surface, introducing students to robotics, sensors, and feedback control systems.

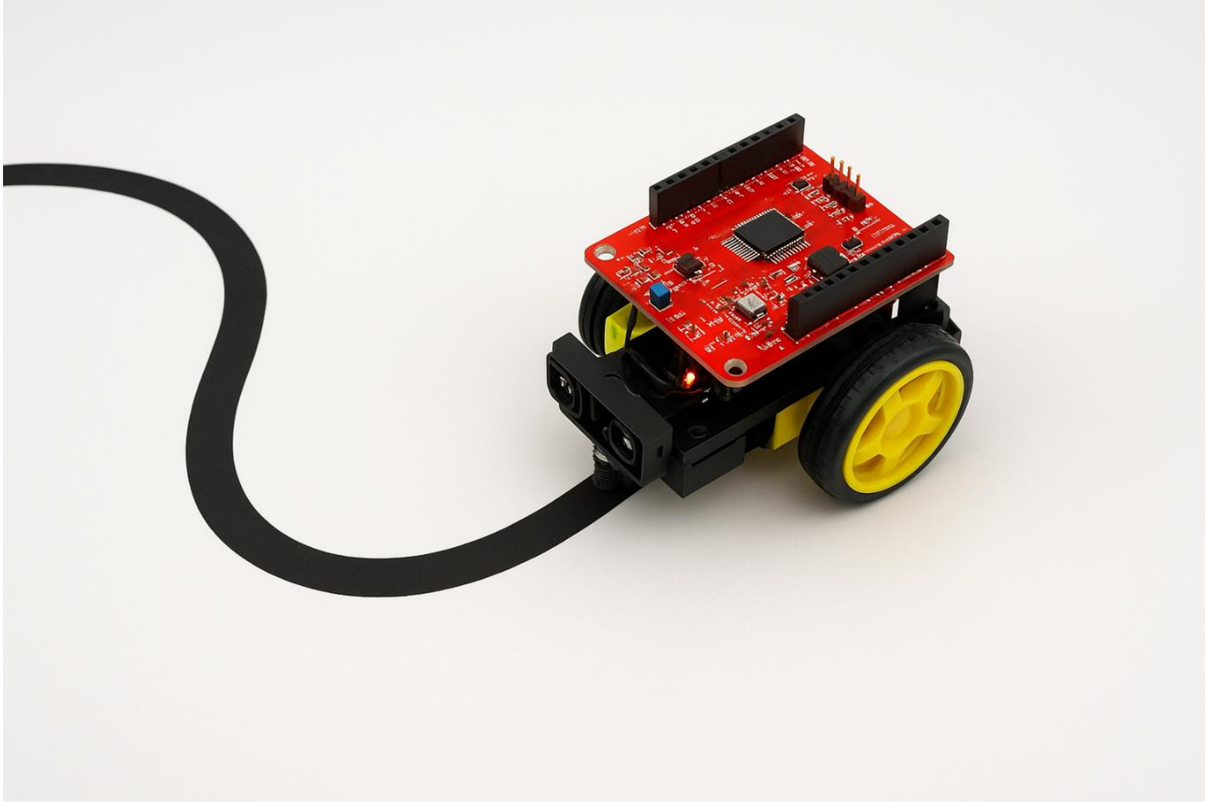
Objectives

1. Introduce students to **IR sensor arrays** and how they detect contrasts (black vs. white).
2. Demonstrate the use of **motor drivers (L293D)** to control DC motors via Arduino.
3. Teach **basic control logic** for autonomous movement.
4. Show the importance of **feedback loops** in robotics for real-time decision-making.
5. Encourage students to participate in **STEM competitions** using autonomous robots.

Goal

By the end of this project, students will:

- Understand how sensors interact with their environment.
- Be able to design, wire, and program a robot that follows a line.
- Learn the basics of control systems (if–else decisions and feedback loops).
- Gain practical experience in robotics competitions and challenges.



Materials Required

- Arduino Uno (1x)
- IR sensor array (3–5 sensors recommended)
- Motor driver IC (L293D) or module
- 2x DC geared motors + wheels
- Robot chassis (with caster wheel or skid)
- Battery pack (Li-ion or AA-based)
- Jumper wires and breadboard

Skills Learned

- **Electronics:** Motor driver wiring, sensor interfacing, power distribution.
- **Programming:** Sensor reading, decision-making with if–else statements.
- **Robotics Concepts:** Feedback loops, basic automation, autonomous navigation.
- **Problem-Solving:** Debugging sensor misreads and tuning motor speeds.

Timeframe

- **Week 1:** Assemble chassis, mount motors, and set up motor driver with Arduino.
- **Week 2:** Mount IR sensors, write initial code for simple left–right control.
- **Week 3:** Test robot on tracks, adjust thresholds, and refine logic.
- **Optional Week 4:** Implement PID control for competition-ready performance.

Step-by-Step Process

1. **Prepare the Chassis**
    - Fix DC motors to the chassis.
    - Attach wheels and balance with a caster wheel/skid.
2. **Mount the Motor Driver (L293D)**
    - Connect motor wires to OUT1–OUT4 pins of the driver.
    - Connect input pins (IN1–IN4) to Arduino digital pins.
    - Power motor driver from the battery pack.
3. **Attach the IR Sensor Array**
    - Fix the array at the front of the robot, close to the ground.
    - Wire sensor outputs to Arduino analog/digital pins.
    - Calibrate sensors so they detect black vs. white properly.
4. **Wire the Power System**
    - Connect battery pack to both Arduino (VIN) and motor driver.
    - Ensure all grounds are connected (Arduino, driver, sensors).
5. **Write and Upload the Code**
    - Read sensor values.
    - If the center sensor detects the line → go straight.
    - If left sensor detects → turn left.
    - If right sensor detects → turn right.
    - Use `if-else` statements to implement logic.
6. **Testing the Robot**
    - Place robot on a simple track (black tape on white floor).
    - Observe movement and adjust sensor thresholds.
    - Fine-tune motor speed balance for smoother turns.

Extensions

- **PID Control:** Implement proportional–integral–derivative logic for precise line following.
- **Speed Optimization:** Adjust code for faster line following without losing accuracy.
- **Race Challenge:** Compete on longer tracks with sharp turns.
- **Maze Solver (Advanced):** Extend robot to handle intersections and make decisions.
- **Bluetooth/WiFi Module (Advanced):** Add remote monitoring and control

---

Assessment Project: Smart Traffic Intersection with Line-Following Car

---

Aim

To integrate traffic light control with autonomous vehicle navigation by designing a system where a line-following robot responds to an Arduino-controlled traffic light system, simulating a **real-world smart traffic intersection**.

Objectives

1. Test student ability to **wire and program LEDs** in a sequence (traffic signal).
2. Apply **sensor-based control** using IR sensors for vehicle navigation.
3. Demonstrate understanding of **timing logic** and **conditional decision-making**.
4. Encourage **integration of multiple subsystems** into a working prototype.
5. Simulate a **real-world scenario** (traffic management + autonomous cars).

Goal

The goal is for students to build a **mini traffic system** where a **line-following robot stops and moves according to LED traffic signals**, showcasing both **basic electronics** and **robotics fundamentals**.

Materials

- Arduino Uno (x2) → one for traffic lights, one for robot
- LEDs (red, yellow, green)
- Resistors (220Ω)
- Breadboard + jumper wires
- IR sensor array (3–5 sensors)
- L293D motor driver module
- 2 × DC motors + robot chassis with wheels
- Battery pack (9V or Li-ion)

Skills Learned

- Circuit design with LEDs
- Programming **delays and sequences**

- Motor control and PWM basics
- Using **IR sensors for path detection**
- Conditional programming (IF–ELSE)
- System integration (traffic light → robot behavior)

Timeframe

2–3 weeks (students already practiced both systems separately; integration requires testing and debugging).

Steps

1. **Build Traffic Light System**
   - Connect red, yellow, green LEDs to Arduino pins.
   - Program a traffic cycle (e.g., Green = 5s, Yellow = 2s, Red = 5s).
2. **Build Line-Following Robot**
   - Assemble robot chassis with DC motors + IR sensor array.
   - Program robot to follow a black line on a white track.
3. **Integrate Traffic Signal with Robot**
   - Place traffic lights over the track (at an intersection).
   - Program robot to **stop if red LED is ON** (using a photodiode/IR sensor to detect red light OR by direct signal from Arduino via wireless/wired connection).
   - Robot moves forward only on green.
4. **Test Scenario**
   - Run both systems together.
   - Robot follows line until it reaches the "traffic light zone."
   - It should obey signals: **stop on red, wait, then continue on green**.

Extensions

- Add a **pedestrian button** that interrupts the cycle and forces red light.
- Add multiple robots on the same track to simulate **traffic congestion**.
- Use a **buzzer** for sound signals (e.g., beep when light changes).
- Expand to a **two-intersection system** with coordinated lights.

Intermediate Level

Project 3: Smart Home Automation

Aim

To design and build a smart home automation system using Arduino/ESP8266 that controls appliances, monitors environmental conditions, and enables remote operation through Wi-Fi.

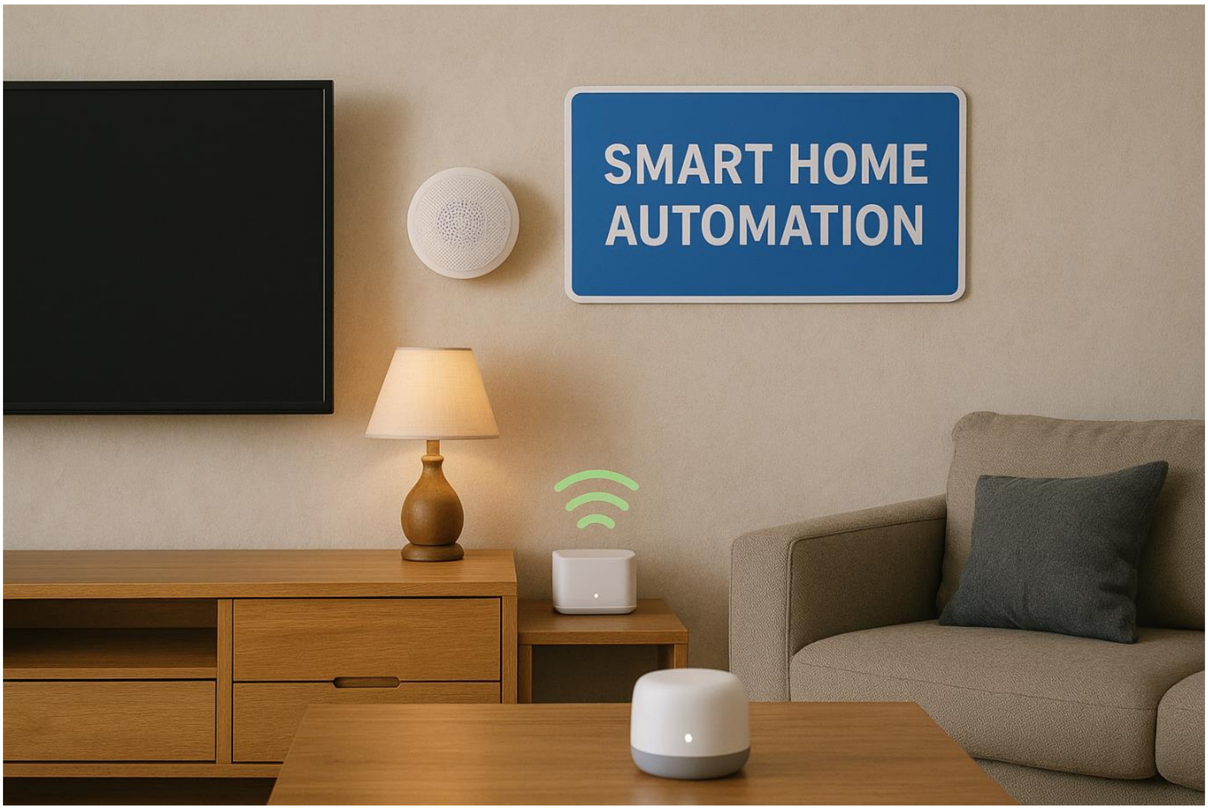
Objectives

1. Introduce students to **Internet of Things (IoT)** concepts and real-world applications.
2. Teach how to use **sensors (temperature, humidity, light)** for environment monitoring.
3. Demonstrate how to control household devices using a **relay module**.
4. Develop skills in **Wi-Fi networking and remote control** of embedded systems.
5. Encourage innovation by extending projects into **smart dashboards and voice assistants**.

Goal

By the end of this project, students will:

- Understand how IoT enables **remote monitoring and automation**.
- Be able to wire and program sensors, relays, and Wi-Fi modules with Arduino/ESP8266.
- Control devices such as lights or fans automatically or remotely.
- Gain practical experience for **smart home and IoT applications** in the real world.



Materials Required

- Arduino Uno or ESP8266 NodeMCU (1x)
- Relay module (2–4 channel, for controlling appliances)
- DHT11 sensor (temperature + humidity)
- Light sensor (LDR or digital light sensor)
- Wi-Fi module (if using Arduino; ESP8266 has Wi-Fi built-in)
- Jumper wires, breadboard, USB cable
- Small appliances (e.g., lamp, fan, LED bulb for demo)

Skills Learned

- **Electronics:** Wiring relays, sensors, and microcontrollers.
- **Programming:** Arduino/ESP8266 coding, sensor data handling.
- **IoT Fundamentals:** Wi-Fi communication, remote device control.
- **Automation Logic:** “If this, then that” (e.g., turn on fan if hot).
- **Innovation & Creativity:** Expanding system to smart dashboards or assistants.

Timeframe

- **Week 1:** Learn relay basics, control lights/appliances with Arduino.
- **Week 2:** Integrate DHT11 and light sensor, display data via Serial Monitor.
- **Week 3:** Add Wi-Fi module for remote access.
- **Week 4:** Test automation rules, finalize prototype, present use cases.

Step-by-Step Process

1. **Set Up Relay Control**
   - Connect relay module to Arduino/ESP8266.
   - Test turning on/off an LED lamp or fan through code.
2. **Add Sensors**
   - Wire DHT11 sensor → Arduino analog/digital pin.
   - Wire light sensor (LDR with resistor divider).
   - Write code to read temperature, humidity, and light levels.
3. **Integrate Wi-Fi**
   - If using ESP8266 → connect to Wi-Fi directly.
   - If using Arduino → connect ESP8266 Wi-Fi module.
   - Test basic communication (e.g., send sensor data to serial or cloud).
4. **Write Control Logic**
   - Example rules:
     - If temperature > 30°C → turn fan ON.
     - If room is dark → turn light ON.
   - Upload and test automation rules.
5. **Enable Remote Access**
   - Host a simple web server on ESP8266.
   - Allow toggling appliances via smartphone browser.
6. **Final Integration**
   - Combine automation + remote control.
   - Test system with multiple devices.

Extensions

- **Smartphone Dashboard:** Create a mobile/web dashboard showing live sensor readings and appliance status.
- **Voice Assistants:** Integrate with Google Home or Alexa for voice-based control.
- **Energy Monitoring:** Add current sensor to track energy consumption.
- **Advanced Cloud IoT:** Send data to platforms like Blynk, Firebase, or Thingspeak.

Project 4: Obstacle-Avoiding Robot

Aim

To design and build an autonomous robot that uses ultrasonic sensing and servo scanning to detect and avoid obstacles in real time, introducing students to robotics navigation and sensor-based decision-making.

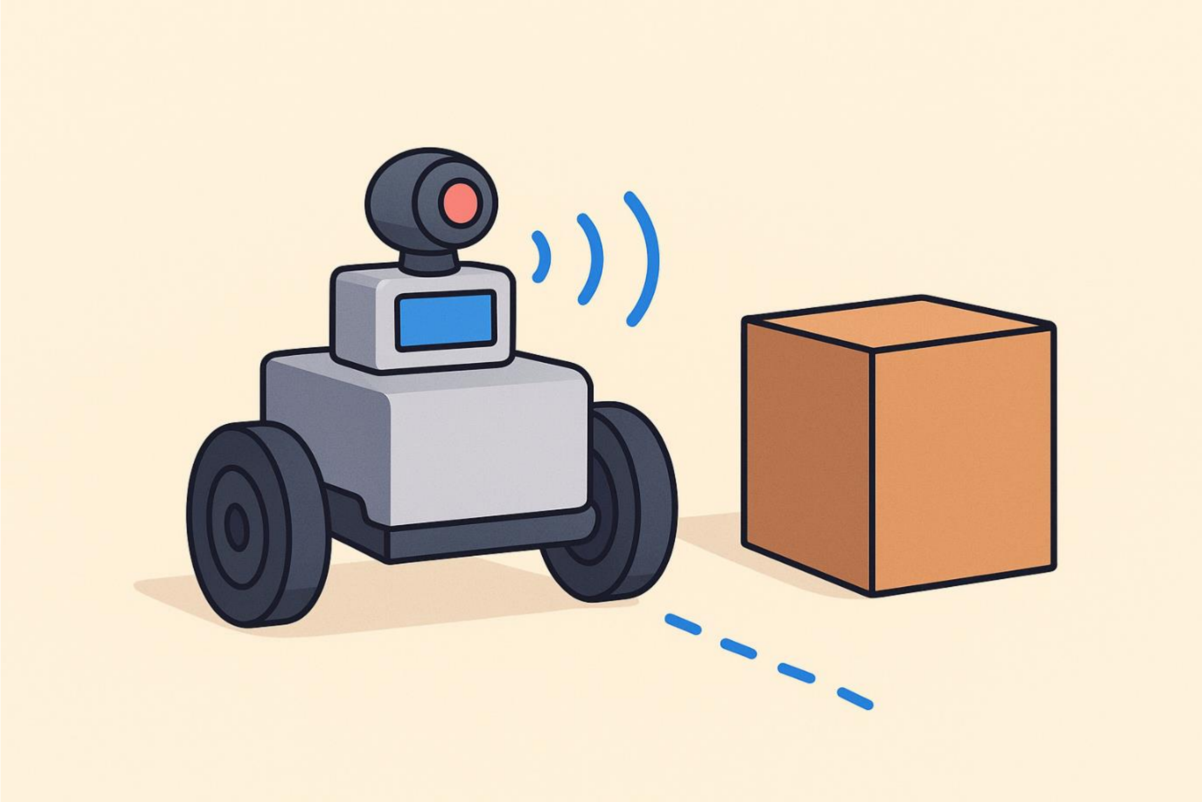
Objectives

1. Teach students how to interface **ultrasonic sensors (HC-SR04)** with Arduino.
2. Introduce the use of a **servo motor** for directional scanning.
3. Demonstrate real-time **robot control logic** for movement decisions.
4. Develop programming skills for **distance measurement and decision-making**.
5. Encourage critical thinking by exploring extensions like **multi-sensor systems** and **AI navigation**.

Goal

By the end of this project, students will:

- Understand how robots sense their environment using **ultrasonic sensors**.
- Be able to build a robot that **detects and avoids obstacles automatically**.
- Learn to integrate **servo scanning with movement logic**.
- Gain foundational skills for **advanced robotics navigation and AI pathfinding**.

- 

Materials Required

- Arduino Uno (1x)
- Ultrasonic sensor (HC-SR04)
- Servo motor (SG90 or MG90S)
- Motor driver (L293D or L298N)
- 2x DC geared motors + wheels
- Robot chassis (with caster wheel/skid)
- Li-ion battery pack (or AA pack)
- Jumper wires, breadboard

Skills Learned

- **Electronics:** Sensor and servo interfacing, motor driver connections.
- **Programming:** Distance measurement, servo control, decision-making logic.
- **Robotics:** Real-time obstacle avoidance, navigation strategies.
- **Problem-Solving:** Debugging sensor readings, fine-tuning servo sweeps.
- **Innovation:** Exploring AI-based navigation and multi-sensor integration.

Timeframe

- **Week 1:** Assemble chassis, mount motors, and wire motor driver.
- **Week 2:** Interface ultrasonic sensor and servo with Arduino.
- **Week 3:** Write and test obstacle-avoidance code (stop, turn, move).
- **Week 4:** Refine servo scanning, test in real-world obstacle courses.

Step-by-Step Process

1. **Prepare the Chassis**
    - Fix DC motors to the chassis and attach wheels.
    - Add a caster wheel/skid for balance.
2. **Mount Motor Driver**
    - Connect motors to motor driver outputs.
    - Connect driver inputs (IN1–IN4) to Arduino pins.
    - Provide motor power via battery pack.
3. **Attach Ultrasonic Sensor + Servo**
    - Mount HC-SR04 on top of a servo motor.
    - Fix assembly to the front of the chassis.
    - Wire servo (signal → Arduino digital pin, power → 5V).
4. **Wire Power System**
    - Battery pack → motor driver.
    - Arduino powered via VIN or USB.
    - Common ground for all components.
5. **Write and Upload the Code**
    - Use servo to sweep ultrasonic sensor left, center, and right.
    - Measure distances at each angle.
    - Logic:
        - If path ahead is clear → move forward.
        - If obstacle ahead → stop and turn to the clearer side.
6. **Testing the Robot**
    - Place robot in a corridor or room with boxes/objects.
    - Observe obstacle detection and turning behavior.
    - Adjust distance thresholds (e.g., stop if < 20 cm).

Extensions

- **360° Detection:** Add multiple ultrasonic sensors around the robot.
- **Path Optimization:** Use algorithms to select best direction.
- **AI Pathfinding:** Implement basic maze-solving or SLAM (Simultaneous Localization and Mapping).
- **Wireless Control:** Add Bluetooth/Wi-Fi for remote monitoring of robot path.
- **Camera Integration (Advanced):** Add a camera module for vision-based navigate

---

Project 5: Bluetooth-Controlled Car

Aim

To design and build an Arduino-based car that can be controlled wirelessly via a smartphone using Bluetooth, introducing students to wireless communication and mobile–robot integration.

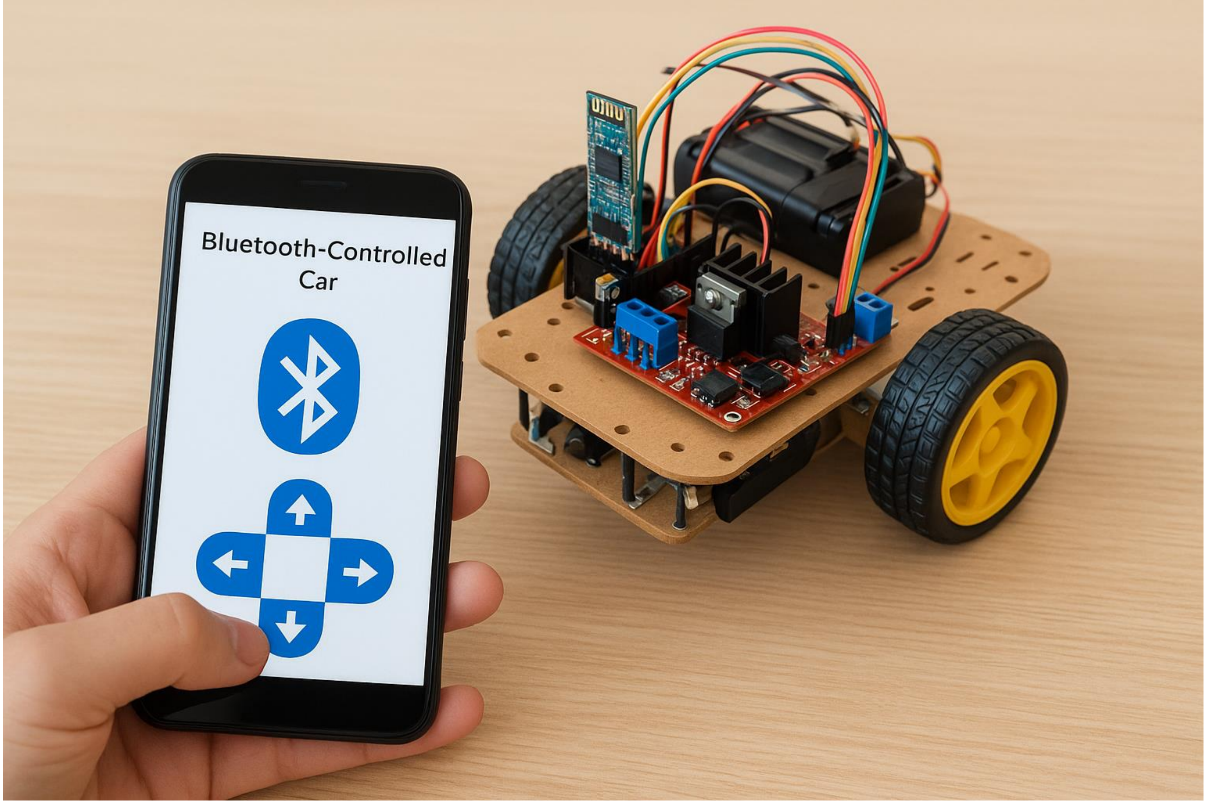
Objectives

1. Teach students how to use the **HC-05 Bluetooth module** with Arduino.
2. Demonstrate **wireless serial communication** between a phone and a microcontroller.
3. Guide students in programming Arduino to **interpret commands** (forward, backward, left, right, stop).
4. Show how to integrate a **motor driver** for controlling robot motion.
5. Encourage creativity by adding **extra features** such as headlights, horns, or voice commands.

Goal

By the end of this project, students will:

- Understand how Bluetooth enables **wireless communication** with embedded systems.
- Be able to pair a smartphone with Arduino via the **HC-05 module**.
- Write code to receive and process **Bluetooth commands**.
- Build a fully functional **Bluetooth-controlled robotic car**.



Materials Required

- Arduino Uno (1x)
- HC-05 Bluetooth Module (1x)
- Motor Driver Module (L293D or L298N)
- 2x DC geared motors + wheels
- Robot chassis with caster wheel/skid
- Battery pack (Li-ion or AA pack)
- Jumper wires and breadboard
- Smartphone with Bluetooth

Skills Learned

- **Electronics:** Motor driver and Bluetooth wiring.
- **Programming:** Serial communication, command parsing.
- **Wireless Technology:** Pairing and communicating via HC-05.
- **Mobile Integration:** Using a smartphone app to control hardware.
- **Innovation:** Extending functionality with custom features (voice, lights).

Timeframe

- **Week 1:** Assemble chassis, mount motors, and wire motor driver.
- **Week 2:** Connect HC-05 Bluetooth module and test communication.
- **Week 3:** Write code to map received commands to robot movements, test with app.

Step-by-Step Process

1. **Assemble the Car Base**
   - Fix motors and wheels to the chassis.
   - Add caster wheel/skid for balance.
2. **Connect Motor Driver**
   - Wire DC motors to driver outputs.
   - Connect driver inputs to Arduino pins.
   - Power motor driver with battery pack.
3. **Integrate Bluetooth Module (HC-05)**
   - VCC → 5V, GND → GND.
   - TX of HC-05 → RX of Arduino.
   - RX of HC-05 → TX of Arduino (use voltage divider if needed).
4. **Pair Smartphone with HC-05**
   - Enable Bluetooth on phone.
   - Search and pair with HC-05 (default password: 1234/0000).
   - Use a controller app (like "Bluetooth Terminal" or custom MIT App Inventor app).
5. **Write and Upload Arduino Code**
   - Use `Serial.read()` to capture commands from phone.
   - Map commands:
     - `F` → Forward
     - `B` → Backward
     - `L` → Left
     - `R` → Right
     - `S` → Stop
6. **Test the Car**
   - Send commands via smartphone app.
   - Verify correct motion response.
   - Adjust motor speed if needed.

Extensions

- **Headlights/Horn:** Add LEDs and a buzzer controlled via Bluetooth commands.
- **Voice Control:** Use Android voice commands mapped to Bluetooth signals.
- **Custom Mobile App:** Build a dedicated control app with buttons and joystick.
- **Speed Control:** Implement PWM to adjust car speed from the phone.
- **GPS Tracking (Advanced):** Add GPS module to log car's movement path.

Assessment Project: IoT-Enabled Smart Rover

Aim

To design and build a **multi-functional robotic rover** that integrates **IoT home automation, obstacle avoidance, and Bluetooth wireless control**, demonstrating a blend of automation, sensing, and mobile communication.

Objectives

1. Test student ability to combine **IoT (ESP8266/Arduino Wi-Fi)** with robotics.
2. Apply **real-time sensing (ultrasonic, servo scanning)** for autonomous navigation.
3. Demonstrate **wireless communication** using Bluetooth and smartphone apps.
4. Enable **dual control modes**: autonomous obstacle avoidance + remote control.
5. Encourage **systems thinking** by integrating sensors, actuators, and wireless modules in one project.

Goal

The goal is for students to build a **hybrid smart rover** that can:

- Be controlled via **Bluetooth smartphone app**
- Switch into **autonomous mode** and avoid obstacles
- Connect to Wi-Fi to **report environmental data (temperature, humidity, light)** and allow remote appliance/LED control (smart home demo).

Materials

- Arduino Uno + ESP8266 (or NodeMCU ESP8266 alone)
- Motor driver (L298N or L293D)
- DC motors + robot chassis with wheels
- Ultrasonic sensor (HC-SR04) + servo motor for scanning
- HC-05 Bluetooth module
- DHT11 sensor (temperature/humidity)
- Light sensor (LDR + resistor)
- Relay module (for controlling a lamp/LED as a smart appliance demo)
- Smartphone (for Bluetooth app + Wi-Fi dashboard)
- Power supply (Li-ion battery or 9V pack)

Skills Learned

- IoT and smart home basics (Wi-Fi data reporting, relay control)
- Wireless communication with Bluetooth (manual control via phone)
- Real-time robotics sensing (ultrasonic scanning for obstacle detection)
- Multi-mode programming (switch between Bluetooth/manual vs. autonomous navigation)
- Systems integration: combining **robotics + IoT + mobile control**

Timeframe

4–6 weeks (requires students to build three subsystems, then integrate and test).

Steps

1. **Build the Rover Base**
    - Mount motors, wheels, chassis, and motor driver.
    - Add power supply.
2. **Add Obstacle-Avoidance System**
    - Mount ultrasonic sensor on a servo for scanning.
    - Program autonomous mode: move forward until obstacle, scan left/right, turn toward free space.
3. **Enable Bluetooth Control**
    - Wire HC-05 to Arduino.
    - Build simple Android app (via MIT App Inventor) to send commands (forward, backward, left, right, stop).
    - Map received commands to motor actions.
4. **Integrate Smart Home IoT Features**
    - Add DHT11 and LDR sensors to Arduino/ESP8266.
    - Send data via Wi-Fi to a dashboard (ThingSpeak, Blynk, or custom webpage).
    - Connect a relay to control a lamp/LED remotely from the same dashboard.
5. **Combine Modes**
    - Use a toggle (via Bluetooth command or Wi-Fi dashboard) to switch between:
        - **Manual Bluetooth Mode** → Rover moves by phone joystick.
        - **Autonomous Mode** → Rover avoids obstacles automatically.
        - **IoT Mode** → Rover collects environment data + remote relay control.
6. **Test System**
    - Drive rover with phone.
    - Switch to autonomous mode and watch it avoid obstacles.
    - Check IoT dashboard for live sensor readings.
    - Turn lamp/LED ON/OFF remotely.

Extensions

- Add a **camera (ESP32-CAM)** for live video streaming to the dashboard.
- Integrate **voice control** (via Google Assistant or Alexa).
- Add **geofencing**: rover can only move inside a defined area (Wi-Fi RSSI + GPS module).
- Expand to **multi-rover IoT system**, with each rover reporting data to a shared dashboard.

Advanced Level

Project 5: Bluetooth-Controlled Car

Aim

To design and build an automated sorting system that uses sensors and motors to identify objects based on color and organize them into separate bins, mimicking real-world industrial automation systems.

Objectives

1. Learn how sensors (color sensor TCS34725) detect and classify objects.
2. Understand conveyor-belt mechanics and integration with servos and DC motors.
3. Develop logical decision-making in Arduino programming (if–else conditions).
4. Gain hands-on experience in automation and industrial robotics concepts.

Goal

The goal is to create a robotic sorting system capable of detecting an object's color and moving it into the correct bin automatically, while preparing students for concepts in industrial robotics and mechatronics.

Materials

- Arduino Mega (for more I/O pins and flexibility)
- TCS34725 color sensor
- Servo motors (for gate sorting mechanism)
- DC motor (to run the conveyor)
- Motor driver module
- Conveyor belt system (can be DIY using rollers, rubber band/belt)
- Supporting frame (wood, acrylic, or 3D-printed parts)
- Power supply (12V recommended)
- Wires, resistors, and breadboard

Skills Learned

- Automation and control system design
- Sensor-based decision making
- Servo and DC motor integration
- Conveyor belt mechanical design
- Basics of industrial robotics

Timeframe

3–4 weeks (including mechanical setup and programming)

Steps

1. **Build the conveyor system**
   - Assemble conveyor using rollers and belt.
   - Mount DC motor to drive the belt.
2. **Mount the color sensor (TCS34725)**
   - Fix it above the conveyor so objects pass under it.
   - Wire it to Arduino Mega (I2C pins).
3. **Add sorting mechanism**
   - Place servos at the end of the conveyor with gates or pushers.
   - Servos should rotate to direct objects into correct bins.
4. **Wire connections**
   - DC motor → motor driver → Arduino.
   - Servo → PWM pins on Arduino.
   - Color sensor → I2C pins.
5. **Program the sorting logic**
   - Read RGB values from the sensor.
   - Use thresholds to classify objects as Red, Green, or Blue.
   - Move servo to push object into the correct bin.
6. **Test with objects**
   - Place colored balls/blocks on the conveyor.
   - Verify correct sorting into bins.

Extensions

- Sort by **size or weight** using ultrasonic or load cell sensors.
- Add an **LCD counter** to display how many items sorted by each category.
- Use **machine learning** for advanced classification (shapes, patterns).
- Scale into a **multi-conveyor system** for complex sorting.

Project 7: Wi-Fi Rover

Aim

To design a Wi-Fi-controlled robotic rover with live video streaming, allowing users to control and monitor the rover remotely using a smartphone or PC.

Objectives

1. Learn wireless communication using ESP32/ESP32-CAM.
2. Gain hands-on experience with live video streaming.
3. Understand integration of motor drivers, DC motors, and camera modules.
4. Explore IoT and real-time control applications.

Goal

The goal is to create a remotely controlled rover capable of transmitting live video while being navigated via Wi-Fi, simulating real-world teleoperation in robotics.\


Wi-Fi
ROVER

Materials

- ESP32 or ESP32-CAM module
- L298N or L293D motor driver
- DC motors (x2 or x4) + wheels and chassis
- Lithium-ion battery pack or rechargeable power supply
- Camera module (if not using ESP32-CAM built-in)
- Smartphone or laptop for control interface
- Jumper wires and connectors

Skills Learned

- IoT basics and wireless control
- Real-time camera streaming over Wi-Fi
- Motor driver + DC motor integration
- Remote control UI development (phone/laptop dashboard)
- Basics of mobile robotics

Timeframe

3–4 weeks

Steps

1. **Assemble rover chassis**
    - Attach DC motors and wheels to chassis.
    - Secure ESP32 board and motor driver.
2. **Wire motors + ESP32**
    - Connect DC motors to L298N driver.
    - L298N → ESP32 GPIO pins.
    - Power both using Li-ion battery pack.
3. **Integrate camera module**
    - Use ESP32-CAM or connect external camera.
    - Position camera for forward-facing view.
4. **Upload Wi-Fi control + streaming code**
    - Flash ESP32 with code that:
        - Creates a Wi-Fi server.
        - Streams live video feed.
        - Accepts motor control commands (forward, backward, left, right).
5. **Build smartphone web interface**
    - Simple webpage hosted on ESP32.

    - Buttons or joystick for rover control.
    - Live camera feed embedded.
6. **Test locally**
    - Connect phone to ESP32 Wi-Fi network.
    - Open rover control webpage.
    - Drive rover and view live video.

Extensions

- Add **obstacle avoidance** using ultrasonic sensors.
- Implement **joystick control** (physical or app-based).
- Create a **telemetry dashboard** (speed, battery status).
- Integrate with **cloud IoT platforms** for remote internet control.
- Add **autonomous navigation** features (line-following + AI pathfinding).

Project 8: Self-Parking Car

Aim

To design and build an **autonomous self-parking robotic car** that uses ultrasonic sensors and servo-based steering to detect parking spaces, plan trajectories, and execute real-world parking maneuvers without human control.

Objectives

1. Introduce students to **Arduino Mega** as a microcontroller for robotics.
2. Learn to integrate **multiple ultrasonic sensors** for obstacle and space detection.
3. Understand **servo-based steering mechanisms** and motor driver integration.
4. Implement **motion planning algorithms** for parallel parking.
5. Gain experience in **sensor fusion and decision-making in robotics**.
6. Explore the fundamentals of **autonomous vehicle technology**.

Goal

The goal is to create a **self-parking car prototype** that can autonomously detect parking spaces, plan a safe trajectory, and park itself in a mini parking lot, demonstrating real-world concepts of **autonomous driving systems**.

Materials

- Arduino Mega 2560
- Ultrasonic sensors (HC-SR04 × 3–4)
- Steering servo motor
- DC motors × 2 (rear drive wheels)
- Motor driver (L298N or L293D)
- 4-wheel chassis with steering linkage
- Battery pack (9–12V)
- Jumper wires, breadboard, mounting tools
- Optional: OLED/LCD display for status messages

Skills Learned

- **Sensor fusion**: using multiple sensors for real-time decisions
- **Motion planning**: implementing parallel parking algorithms
- **Embedded systems**: servo + DC motor control with Arduino
- **Real-world robotics applications**: autonomous vehicle basics
- Debugging and refining robotics logic in physical systems

Timeframe

4–6 weeks (structured with progressive weekly milestones)

Steps

1. **Assemble the car chassis**
    - Mount DC motors on the rear wheels.
    - Attach servo for steering the front wheels.
    - Fix ultrasonic sensors (front, side, and rear).
2. **Wire electronics**
    - Connect DC motors to motor driver.
    - Connect servo motor to PWM pin.
    - Wire ultrasonic sensors (Trig + Echo pins to Arduino Mega).
    - Power system with 9–12V battery pack.
3. **Test sensors individually**
    - Write Arduino sketches to measure distances.
    - Print values to Serial Monitor.
    - Verify reliable front/side/rear detection.
4. **Test motion control**
    - Program Arduino to move forward, reverse, turn left/right.

    - Fine-tune servo angles for realistic steering.
5. **Implement space detection**
    - Car drives forward along a mock "street".
    - Side sensors measure open space length.
    - If space ≥ car length + margin → mark as valid parking spot.
6. **Parallel parking algorithm**
    - Reverse with wheels turned fully right.
    - Straighten wheels, continue reversing.
    - Turn wheels fully left and move forward to align.
    - Stop when fully parked inside spot.
7. **Testing & debugging**
    - Use a cardboard mini parking lot.
    - Adjust sensor thresholds, timing, and angles for accuracy.

Extensions

- Add **multiple parking modes**: parallel, perpendicular, or garage parking.
- Integrate an **OLED/LCD screen** to display status messages (e.g., "Searching… Parking… Parked").
- Add a **camera** for vision-guided parking.
- Integrate **Bluetooth/Wi-Fi** for monitoring or remote override.
- Apply **basic AI/ML models** for space recognition and optimized trajectories.

Assessment Project (Advanced Level): Smart Autonomous Rover with Sorting Docking Station

Concept

A Wi-Fi–controlled rover that can **navigate autonomously into a docking station**, detect and **sort colored objects** it carries, and provide **live video streaming** and **remote monitoring**.
This combines **automation (sorting), IoT & wireless control (Wi-Fi rover), and sensor fusion/motion planning (self-parking car).**

Materials

- **Core Controller**: Arduino Mega + ESP32-CAM (Arduino handles sorting & motion, ESP32 handles Wi-Fi and streaming)
- **Sensors**:
    - TCS34725 Color Sensor (object sorting)
    - Ultrasonic Sensors (HC-SR04, for parking/docking & obstacle detection)
- **Actuators**:
    - Servo motors (sorting gate & steering)
    - DC motors + motor driver (locomotion & conveyor belt)
- **Other Hardware**:
    - Conveyor system / mini-docking station
    - LEDs + buzzer (status indication)
    - Frame/chassis + battery pack

Skills Reinforced

- **Automation & Mechatronics** (sorting machine logic, conveyor integration)
- **IoT & Wireless Robotics** (ESP32 Wi-Fi control + live video streaming)
- **Sensor Fusion & Motion Planning** (self-parking into docking bay)
- **System Integration** (Arduino + ESP32 coordination)

Timeframe

5–6 weeks (modular build: rover → docking station → integration → testing)

Steps

1. **Rover Base Build**
   - Assemble chassis, mount motors, connect ESP32 for Wi-Fi control.
   - Test basic forward/backward/turning commands via smartphone web app.
2. **Add Live Video Streaming**
   - Enable ESP32-CAM streaming to phone/laptop dashboard.
   - Add joystick/arrow key control overlay.
3. **Autonomous Navigation (Self-Parking Mode)**
   - Mount ultrasonic sensors front + sides.
   - Program rover to detect a "docking station" and align/park automatically.
4. **Docking Station with Conveyor + Sorting**
   - Build a mini conveyor belt at docking station.
   - Mount TCS34725 color sensor above it.
   - Add servo gate that sorts objects into bins based on color.
5. **Integration**
   - Rover carries small colored objects → drives to docking station → parks itself → drops objects on conveyor → sorting begins.
6. **Testing & Optimization**
   - Test remote control + autonomous mode switching.
   - Calibrate sorting accuracy, docking alignment, and video latency.

Extensions

- Add **object weight sensor** for multi-factor sorting (color + weight).
- Implement **AI vision** with ESP32-CAM (object detection instead of just color).
- Add **telemetry dashboard**: battery level, number of objects sorted, parking success rate.
- Multi-rover system → 2 rovers sharing the same docking/sorting station.

Expert Level

Project 9: Robotic Exoskeleton Glove

Aim

To design and build a wearable robotic glove that mimics hand and finger movements using flex sensors and servo motors, bridging biomechanics with robotics for applications in rehabilitation, teleoperation, and human-robot interaction.

Objectives

1. Understand how **flex sensors** detect finger bending and translate it into digital signals.
2. Learn how **servo motors** can replicate human finger movements through tendon-based actuation.
3. Explore **wearable robotics** for assistive technologies, rehabilitation, and remote control.
4. Develop skills in **Arduino programming** for sensor-to-actuator mapping.
5. Introduce concepts of **human-robot interfaces** and gesture-based control systems.

Goal

The goal is to create a robotic glove system that translates human hand movements into robotic hand movements in real time, enabling practical use in rehabilitation, prosthetics, and telepresence robotics.

Materials

- Arduino Nano (compact and lightweight for wearables)
- Flex sensors (5x, one per finger)
- Servo motors (5x, one per finger joint of robotic hand)
- 3D-printed robotic hand (or pre-made robotic hand kit)
- Glove (preferably fabric-based for easy sensor mounting)
- Tendon material (nylon fishing line or similar)
- Resistors (for sensor voltage dividers)
- Breadboard and jumper wires
- External 5–6V power supply for servos

Skills Learned

- Wearable robotics design
- Biomechanics and movement mapping
- Human-robot interaction concepts
- Arduino programming (analog sensor inputs and servo outputs)
- Basic mechatronics and tendon-driven actuation

Timeframe

5–6 weeks (longer due to mechanical design and calibration)

Steps

1. **Prepare the glove**
   - Mount 5 flex sensors along the glove’s fingers.
   - Wire each sensor with resistors in voltage divider circuits.
2. **Build the robotic hand**
   - 3D-print or assemble the robotic hand.
   - Install servo motors at each finger joint.
   - Use tendons (nylon thread) to connect servos to finger tips.
3. **Connect electronics**
   - Flex sensors → Arduino Nano analog inputs (A0–A4).
   - Servos → Arduino digital PWM pins.
   - Power → external supply for servos (shared ground with Arduino).
4. **Program glove-to-robot mapping**
   - Read flex sensor values (finger bend).
   - Map analog values to servo angles.
   - Write servo control logic so robotic fingers mimic human fingers.
5. **Test calibration**
   - Bend each finger slowly.
   - Observe robotic hand follow motion.
   - Adjust mapping for smoother response.

Extensions

- Add **Bluetooth (HC-05)** for wireless operation.
- Integrate **haptic feedback** using vibration motors to simulate touch.
- Add **IMU sensor** (gyroscope/accelerometer) for gesture recognition.
- Develop **rehabilitation applications** (track patient progress).
- Integrate into **VR/AR environments** for immersive experiences.

Project 10: AI Vision Rover (Capstone Project)

Aim

To design and implement an intelligent rover powered by computer vision and AI, capable of perceiving its environment, making autonomous navigation decisions, and serving as an introduction to **AI-driven robotics** and **real-world machine learning applications**.

Objectives

1. Introduce students to **Raspberry Pi** as a robotics and AI platform.
2. Learn **computer vision basics** using the Pi Camera and OpenCV.
3. Understand the integration of **motors, sensors, and vision systems** into a single robot.
4. Gain hands-on experience in **Python programming** and AI model deployment.
5. Develop problem-solving skills in debugging real-time robotic systems.
6. Explore extensions into **object detection, tracking, and navigation using AI models**.

Goal

The goal is to create a **fully autonomous AI-powered rover** that can process live camera input to recognize objects, follow paths, or avoid obstacles, demonstrating the integration of **robotics, AI, and computer vision** in a practical capstone project.


AI VISION ROVER

Materials

- Raspberry Pi 4 (2GB or higher recommended)
- Pi Camera Module (v2 or HQ camera)
- Motor driver (L298N or similar H-bridge)
- Robot chassis with DC motors and wheels
- Power bank or Li-ion battery pack
- Jumper wires, breadboard (if prototyping connections)
- MicroSD card (16GB+) with Raspberry Pi OS
- Optional: ultrasonic sensor (for extra obstacle avoidance)

Skills Learned

- AI & computer vision (OpenCV, image processing)
- Python programming for robotics
- Integration of hardware (motors, camera, microcontroller)
- Real-time decision-making in autonomous systems
- Foundations of **machine learning deployment** on embedded devices
- Introduction to **ROS (Robot Operating System)** for advanced robotics

---

Timeframe

6–8 weeks (capstone-level project requiring coding, hardware integration, and AI experimentation)

Steps

1. **Assemble the rover chassis**
    - Mount DC motors, wheels, motor driver, and Raspberry Pi.
    - Secure the Pi Camera in a front-facing position.
2. **Set up Raspberry Pi environment**
    - Install Raspberry Pi OS.
    - Set up Python, OpenCV, and required libraries.
    - Enable camera module in Pi configuration.
3. **Motor + camera integration**
    - Write Python scripts to control rover motors via GPIO pins.
    - Capture live video feed from the Pi Camera.
    - Test basic forward/backward movement while streaming video.
4. **Implement computer vision**
    - Use OpenCV to detect colors, shapes, or objects.
    - Program the rover to react (e.g., follow a colored object, stop at obstacles).

5. **AI model training/testing**
    - Collect image datasets with the Pi Camera.
    - Train a simple classifier/detector (e.g., traffic signs or object recognition).
    - Deploy the trained model onto the Pi for real-time inference.
6. **Integration & debugging**
    - Combine motor control + AI vision into one program.
    - Test in real-world environments (hallways, tracks, or outdoor areas).

Extensions

- Implement **object following** (e.g., follow a red ball or person).
- Add **traffic sign recognition** for smart navigation.
- Integrate **ROS (Robot Operating System)** for modular robotics control.
- Use **YOLO or TensorFlow Lite models** for real-time object detection.
- Add a **web dashboard** to stream live camera feed and send remote commands.
- Upgrade with **autonomous path planning** and multi-sensor fusion (AI + ultrasonic + IMU).

Assessment Project (Expert Level): Vision-Guided Rover Controlled by Robotic Exoskeleton Glove

---

Concept

Build a rover that can be **remotely controlled using a wearable robotic glove** (flex sensors for hand gestures), but also has the ability to switch to **AI autonomous vision mode** (object following, traffic sign recognition, obstacle avoidance).

This project merges **wearable robotics (glove), AI computer vision (rover), and advanced system integration** into a single system.

Materials

- **Glove Side (Controller)**
  - Arduino Nano
  - Flex sensors (finger bends → mapped to control commands)
  - Servo motors (for haptic feedback or response vibration motor)
  - Bluetooth module (HC-05 or ESP32 for wireless communication)
  - Wearable glove frame
- **Rover Side (Executor)**
  - Raspberry Pi 4
  - Pi Camera module (for vision tasks)
  - Motor driver + DC motors + robot chassis
  - Ultrasonic sensor (extra obstacle detection)
  - Battery + power management

Skills Reinforced

- **Wearable Robotics & Biomechanics** (gesture sensing, glove-to-device mapping)
- **Wireless Communication** (Bluetooth/ESP32 between glove and rover)
- **AI Computer Vision** (OpenCV: object following, sign detection, basic ML integration)
- **Advanced Robotics Integration** (manual + autonomous hybrid control)
- **Human–Robot Interaction** (gesture-based interface + haptic feedback)

Timeframe

7–9 weeks (modular stages: glove → rover → AI → integration → final demo)

Steps

1. Glove Input System

- Mount flex sensors on glove fingers.
- Program Arduino Nano to map bends to directional commands (e.g., fist = stop, index forward = move forward, tilt = turn).
- Send commands via Bluetooth to Raspberry Pi.

2. Rover Base with Remote Control

- Assemble rover chassis with Pi 4 and motor driver.
- Test receiving glove commands over Bluetooth and mapping them to movement.

3. Add AI Vision Layer

- Install OpenCV on Raspberry Pi.
- Implement **basic object detection** (e.g., follow a red ball).
- Implement **traffic sign recognition** (simple CNN trained on stop/go signs).
- Add **autonomous navigation mode** (Pi decides rover motion independent of glove).

4. System Integration

- Build a **dual-mode control system**:
  - Manual Mode → Rover follows glove commands.
  - AI Mode → Rover uses camera & ML model to drive itself.
- Create a **mode switch gesture** (e.g., thumb + index pinch → toggle AI/manual).

5. Feedback & Safety

- Add haptic feedback (servo vibration/buzzer in glove when rover detects obstacle).
- Ensure safety override (clench fist = emergency stop).

6. Testing & Calibration

- Test glove accuracy for gesture recognition.
- Test rover's vision in multiple environments (indoor, outdoor, different lighting).
- Optimize wireless latency and response speed.

Extensions

- Add **gesture + voice hybrid commands** (glove + speech input).
- Integrate **ROS** on Raspberry Pi for modular AI robotics architecture.
- Add **IMU sensor** on glove for 3D hand motion tracking.

- Enable **data logging + performance metrics** (gesture recognition accuracy, AI detection accuracy).

Final Demonstration

Student must demo:

1. Glove control of rover (basic gestures).
2. AI vision following a target object.
3. Smooth transition between **manual glove control ↔ AI mode**.
4. Safety stop gesture + haptic feedback.

This **Assessment Project** forces mastery of:

- Hardware–software integration across two systems (Arduino + Raspberry Pi).
- Human–robot interaction design.
- AI vision + robotics in real-world conditions.
- Robustness in switching between human and AI control.

Assessment Projects for STEM/Robotics Curriculum

Beginner Level

Assessment Project A1: Smart Pedestrian Crosswalk System

(Based on Project 1: LED Traffic Light + Project 2: Line-Following Robot)

Concept: Build a traffic intersection where a **line-following robot car obeys LED traffic lights** and a pedestrian button.

Materials: Arduino Uno, LEDs, resistors, IR sensors, motor driver, DC motors, chassis, push button.

Skills Reinforced: Electronics, sequencing, sensors, real-world traffic simulation.

Timeframe: 2–3 weeks

Steps:

1. Build traffic light system with pedestrian button.
2. Program a line-following car to stop when the light is red.
3. Sync car movement with light cycle.

Extensions: Add buzzer for visually impaired pedestrians, test with multiple cars.

Final Demo: Mini traffic crosswalk with robot cars + pedestrian control.

Intermediate Level

Assessment Project A2: IoT-Enabled Obstacle-Avoiding Car

(Based on Project 3: Smart Home Automation, Project 4: Obstacle-Avoiding Robot, Project 5: Bluetooth Car)

Concept: A car that can be controlled via **Bluetooth app**, but also avoids obstacles and can be remotely toggled via **Wi-Fi automation**.

Materials: Arduino Uno + ESP8266, HC-05 Bluetooth module, ultrasonic sensor, motor driver, chassis, smartphone app.

Skills Reinforced: IoT, wireless comms, sensor-driven control, mobile integration.

Timeframe: 3–4 weeks

Steps:

1. Build Bluetooth-controlled car.
2. Add ultrasonic-based obstacle avoidance.
3. Enable Wi-Fi toggle to switch modes via phone or web dashboard.

Extensions: Voice-controlled driving, integrate into home automation (garage entry).

Final Demo: Drive car with app → switch to auto-avoidance mode → control via Wi-Fi.

Advanced Level

Assessment Project A3: Intelligent Sorting & Delivery Rover

(Based on Project 6: Sorting Machine, Project 7: Wi-Fi Rover, Project 8: Self-Parking Car)

Concept: A **Wi-Fi-controlled rover with a conveyor arm** that can sort colored items, then **park itself in a docking station**.

Materials: ESP32, color sensor, servo motors, conveyor, motor driver, ultrasonic sensors, chassis.

Skills Reinforced: IoT, automation, navigation, mechanical control.

Timeframe: 4–6 weeks

Steps:

1. Build sorting conveyor with color sensor + servos.
2. Mount system on Wi-Fi-controlled rover.
3. Program rover to self-park after sorting task.

Extensions: Add camera for remote monitoring, telemetry dashboard.

Final Demo: Rover collects/sorts items → delivers → self-parks.

Expert Level

Assessment Project A4: Vision-Guided Rover with Robotic Glove Control

(Based on Project 9 + Project 10, already designed)

Concept: Rover can be controlled by **wearable robotic glove gestures**, but also switch to **AI vision autonomy**.

Materials: Arduino Nano + flex sensors (glove), Raspberry Pi + Pi camera (rover), motors, Bluetooth/ESP32 comms.

Skills Reinforced: Human–robot interaction, AI vision, multimodal control.

Timeframe: 7–9 weeks

Steps:

1. Build glove for gesture → rover mapping.
2. Build Pi-based rover with camera + motors.
3. Integrate dual-mode control (manual glove vs. AI).

Extensions: Add haptic glove feedback, ROS-based autonomy.

Final Demo: Glove → rover control; switch to AI → autonomous vision navigation.

Assessment Projects Matching Each Core Project Individually

Assessment Project A5 (for Project 1 – Traffic Light): Interactive Traffic City

- Multiple intersections with smart LEDs and pedestrian push buttons.
- Cars (manual or toy cars) must follow rules.
- Assessment: Correct sequencing, pedestrian safety.

Assessment Project A6 (for Project 2 – Line Follower): Warehouse Delivery Robot

- Line-following robot delivers goods across warehouse tracks.
- Must stop at "loading/unloading zones" marked by IR markers.
- Assessment: Accuracy in navigation and stop zones.

Assessment Project A7 (for Project 3 – Smart Home): IoT Weather Dashboard

- Sensors send data (temp, humidity, light) to Wi-Fi dashboard.
- Lights/fans/appliances toggle automatically.
- Assessment: Correct automation, remote control reliability.

Assessment Project A8 (for Project 4 – Obstacle Robot): Maze-Solver Robot

- Robot must navigate maze using ultrasonic sensors.
- Bonus: Memory-based optimization (learn shortest path).
- Assessment: Efficiency in solving maze.

Assessment Project A9 (for Project 5 – Bluetooth Car): Smartphone Racing Challenge

- Build an app-controlled car.
- Students race cars on marked tracks.
- Assessment: Responsiveness, design creativity, stable app control.

Assessment Project A10 (for Project 6 – Sorting Machine): Recycling Station

- Sorting machine separates plastic, metal, paper (simulated with colors).
- Counts items on LCD display.
- Assessment: Sorting accuracy + system reliability.
