**Assessment type (🗹):**

Questioning (Oral/Written)

Practical Demonstration

3rd Party Report

Other – Project/Portfolio (*please specify)*

**Assessment Resources:**

|  |
| --- |
| * Arduino Uno or equivalent microcontroller/SBC * Sensors (optional, or equivalent for simulation) * Access to cloud platform (e.g., ThingSpeak, AWS IoT, Arduino Cloud) * Computer with appropriate development environment * GitHub account * Circuit components (resistors, wires, breadboard, etc.) * Power supply options (battery pack, AC adapter) * Internet connectivity |

**Assessment Instructions:**

|  |
| --- |
| This assessment requires you to design, develop, and test an IoT system with a corresponding dashboard for NFT Hydroponics. Complete the following tasks: Task 1: Scope Design Options You will need to identify key requirements and clarifications sought from the assessor (acting as the client) or client during scoping, collecting users’ stories and identifying stakeholders to achieve requested outcomes.   * Identify 3-5 key requirements from the NFT Hydroponics scenario, interview with customer/lecturer and user stories. * Create 2 user personas representing different stakeholders (e.g., system owner, maintenance technician, remote monitor) * Develop a requirements prioritisation matrix (Must-Have, Should-Have, Could-Have) * Briefly outline any questions you had for the lecturer/client and their responses.   **Deliverables Timeline:**   * Create a project timeline with major milestones and dependencies  Task 2: Framework Selection Based on the scenario, detail a development and justify a framework that includes:   * Power source: [Battery / External power supply] * Programming language: [Programming language suitable for the microcontroller platform] * Communication protocol: [Serial communication / I2C / SPI] for data transfer between the sensor(s) and the microcontroller/SBC * Network requirements and communication protocols  Task 3: Evaluate IoT and Network Design Options. **Presentation/Report**  **For a Presentation:**   * 10-15 slides * 10-minute presentation duration * Include live demonstration of prototype (5-10 minutes) * Q&A session (5-10 minutes)   The presentation will be delivered covering the following key points (slides):   * Introduction and project vision for the IoT system. * Explain the hardware connections, data flow, and any signal conditioning or preprocessing required. * Justify the sensors chosen and explain how the selected sensor(s) will be integrated with the microcontroller/SBC to achieve the desired monitoring/control function. * Communication Protocols: Discuss the chosen communication protocol for data transfer between the sensor and the microcontroller/SBC. * Data Processing: If using a cloud platform, explain how data will be transmitted and visualized for remote monitoring otherwise, explain data collection and storage. * Power Management: Explain the power source and considerations. * Regulatory Compliance: Briefly mention adherence to safety standards if required. * Highlight the benefits of the proposed system, such as preventing overflow or water waste based on user stories collected. * Error handling approach and testing methods used. * Conclusion: Summarise the key features and advantages of your system.   This can be as a power point that is 10-15 slides and up to 8-minute delivery (which can be recorded) or a report which should be approximately 1,000-1,500 words.  **For a Report:**  **The report covers the same content as the presentation**   * Total word count: 1000 1500 words (excluding code snippets and appendices) * Appendices for technical details, code samples, etc. (as needed)  Task 4: Prototype The prototype can be built using real hardware or simulation software but **must** me a minimal viable working solution. Where sensors are not available, they **must** be simulated with either mock inputs or simulated hardware, e.g. An MCU/SBC that outputs an analogue or digital signal to the prototype. This can also be software based on the same device as the prototype   * Prototype Type: Specify whether the prototype is hardware-based or simulated.   **Minimum Functionality Requirements:**   * Must successfully read from at least 2 sensor types * Must transmit data to a cloud platform or local server * Must implement basic error handling and logging   **Testing Scenarios:**   * Normal operation test case * Error condition test cases (sensor failure, network disconnection) * Performance testing under load * Battery life (if applicable)   **Documentation Requirements:**   * Hardware schematic and wiring diagram * Software architecture diagram * Installation and setup guide   **Bill of Materials:**   * Complete list of components with costs * Alternative components for cost optimisation * Sourcing information * Code Repository: Provide a link to a public repository where the code for the prototype is stored.  Task 5: IoT Dashboard Your designed dashboard should effectively communicate sensor data and provide functionalities that directly benefit the user based on the scenario and their user stories.    **Required Visualizations:**   * Real-time data display using gauges or charts * Historical data trends (hourly, daily, weekly) * System status indicators   **Alerting/Notification System:**   * Threshold-based alerts * Notification methods (email, SMS, push notifications) * Alert history  Submission Guidelines and Required Submissions All students must submit the following through Blackboard:  **Project Files:**   * Complete source code for the prototype * Circuit diagrams and schematics * Dashboard configuration files * Documentation files   **GitHub Repository:**   * A public GitHub repository containing all project code * Repository must include a comprehensive README.md * Repository link must be included in the submission template   **Written Report OR Presentation Slides:**   * If submitting a report: 1,000-1,500 words (excluding code snippets and appendices) * If submitting slides: 10-15 slides in PDF format * Must address all required tasks (1-4)   **Supporting Materials:**   * Hard copy of presentation transcript or script (if presenting) * Video demonstration of working prototype (5 minutes maximum)  Submission Checklist Before submitting, ensure you have:   1. Completed all required tasks (1-4) 2. Created a working prototype (hardware or simulated) 3. Set up a functional dashboard 4. Written comprehensive documentation 5. Created a public GitHub repository with all code 6. Prepared report or presentation slides 7. Included hard copies of all presentation materials |

**Assessment Instrument:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Student Information**  **Name:** **Student ID:** 20078332 **Email:** [**hcshah26@hotmail.com**](mailto:hcshah26@hotmail.com)  [**20078332@tafe.wa.edu.au**](mailto:20078332@tafe.wa.edu.au)  **Submission Type**   * [ ] Report (1,000-1,500 words) * [ ] Presentation (10-15 slides)   **Task 1: Scope Design Options**  **Key Requirements Identified:**   1. Real-time monitoring: Ability to monitor the water flow and reservoir levels in real-time. 2. Automated Alerts: System should trigger alerts for critical conditions like low water levels or overflow. 3. Remote Control: Capability to control the reservoir water pump remotely via the dashboard. 4. Data Logging: Maintain historical data for analysis and system optimisation.   **User Personas Created:**   * **System Administrator**: Oversees the entire hydroponic system, ensures optimal performance, and responds to alerts. * **Maintenance Technician**: Handles physical maintenance tasks, responds to system alerts and performs routine checks.   **Requirements Prioritisation Matrix:**   |  |  | | --- | --- | | **Requirements** | **Priority** | | Real-time monitoring | Must have | | Automated alerts | Must have | | Remote control | Should have | | Data logging | Could have |   **Client Questions and Responses:**   * **Q. What are the current issues you would like us to resolve?**   + **A.** We have a few major setbacks on the hydroponic setup:     - The pump is very noisy and at times when we have students in the lab, we need to shutdown the motor temporarily.     - If the water level in the reservoir tank go below the pump, then the pump will blow out.     - Sometime there is a leak on the trough, and it would be very helpful to be alerted.     - We would also like to monitor the water flowing out and back into the reservoir tank. * **Q. Should the system support mobile notifications?**   1. A. Yes integrating mobile notifications would enhance responsiveness to critical alerts * **Q. Is there any preferred cloud platform?**   1. A. No, we are open to suggestions. * **Q. Would you like us to monitor anything else?**   1. A. Yes, if there a way to monitor fertiliser level in the water would be very handy. * **Q. Would you want us to just monitor the fertiliser levels or do anything else with it?**   1. A. No, for now just being able to monitor will be great. Thanks!   **Project Timeline:**   |  |  | | --- | --- | | **Milestone** | **Timeline** | | Requirement gathering | Week 1 | | System design | Week 2 | | Prototype development | Week 3 - 4 | | Testing and validation | Week 5 | | Final report and presentation | Week 6 |   **Task 2: Framework Selection**  **Power Source:**   * Selected option: External power supply (USB or adapter) * Justification: Due to the nature of the hydroponic setup, a continuous feedback process (monitoring sensors and maintianing communication with cloud) is the ideal solution, we chose to go with an external power supply option and not battery operated.   **Programming Language:**   * Selected language: C++ using Arduino framework * Justification: The ESP32 microcontroller used in the simulation is fully compatible with Arduino IDE and C++ programming. The Arduino ecosystem offers extensive libraries and community support, particularly for:   + GPIO handling   + Interrupts   + MQTT communication   + Wi-Fi connectivity   + Real-time debugging via Serial Monitor   C++ allows low-level control of hardware, which is essential for handling fast interrupts from flowmeters and accurate debouncing for float switches.  **Communication Protocols:**   * Local device protocol: Digital GPIO for flowmeters and float switches * Justification:   + Flowmeters are connected to GPIO pins to capture the high frequency pulses reliably   + Float switches are connected to digital input pins with interrupts for responsive state changes * Wireless protocol: MQTT over Wi-Fi * Justification: MQTT is ideal for IoT applications due to its lightweight nature and low bandwidth consumption. It is supported by Adafruit IO and ensures reliable message delivery with minimal overhead. The ESP32’s built-in Wi-Fi is used to connect to the Adafruit IO broker service (io.adafruit.com) where:   + Sensor data is published to feeds.   + Pump control commands are subscribed and acted upon in real-time.   **Network Requirements:**   * Connectivity type: Wi-Fi (using the onboard module of ESP32-S2) * Bandwidth requirements: Low - MQTT transmits payloads (e.g. flowmeter rates, binary states for switches) * Security considerations:   + Adafruit IO supports secure MQTT over TLS (port 8883), although this project uses the non-SSL port 1883 for Wokwi simulation purposes.   + User credentials (Username + AIO key) are required for broker authentication.   + In real-world deployment, SSL and stronger authentication methods would be recommended.   **Task 3: Evaluate IoT and Network Design Options**  **Delivery Format:**   * [ ] Presentation (10-15 slides, 10-minute duration) * [ ] Report (1,000-1,500 words)   **Introduction and Project Vision**  The goal of this project is to simulate and implement a hydroponic monitoring system that ensures reliable water circulation and prevents water related failures. The system automates monitoring of inflow and outflow water using flowmeter, and detect conditions such as reservoir water level and trough pipe overflow using float switches. All data is transmitted to Adafruit IO for real-time visualisation and remote control of the water pump. This supports proactive maintenance, system automation and efficient water anagement.  **Hardware Connections and Data Flow:**   * Component interconnections:   + Flowmeters – Simulated by generating square wave pulses connected to GPIO pins D3 and D4   + Float switches     - D6 – Horizontal Float Switch for Reservoir warning level     - D7 – Horizontal Float Switch for Reservoir critical level     - D8 – Vertical Float Switch for Pipe overflow warning   + Pump Control Switch: D5 for manual override   + LEDs: D9 (Pump), D12 (Reservoir warning), D13 (Reservoir critical), D14 (Pipe overflow)   + Potentiometers: Analog inputs A1 and A2 to simulate variable flow rates. * Data Flow:   + Interrupts capture pulses from flowmeters to measure water volume   + Sensor readings update internal states   + Data is published via MQTT to Adafruit IO every 30 seconds or when state changes. (We chose 30 seconds update as the Hydroponic system will be able to manage that level of delay, if any urgent action needs to be taken, the interrupts will handle them.) * Signal conditioning/preprocessing:   + Debouncing logic is implemented in software to prevent false triggers from the float switches (push buttons for simulation).   + Edge detection via interrupts ensures responsive and efficient signal handling.   + Flowmeter pulses are counted and converted to flow rate using a calibrated litresperpulse value (7.5L / pulse assumed).   + Critical sensor states automatically disable the pump to prevent from system damage.   **Sensor Selection and Integration:**   * Selected sensors:   + Flowmeters simulators to measure the inflow and outflow rate of water via pulse frequency.   + Float switches (push buttons) simulate digital HIGH / LOW levels to indicate warning, critical and overflow states. * Integration approach:   + All sensors are simulated via GPIO and processed using hardware interrupts.   + Each float switch has a dedicated ISR to toggle states and trigger MQTT updates.   + Pump control logic checks critical / overflow states and disables the pump to prevent flooding or system failure. * Justification:   + Simple digital sensors reduce complexity and cost.   + Interrupt based design avoids continuous polling and is highly responsive.   **Communication Protocol Details:**   * Protocol selection:   + Sensor to Microcontroller: GPIO (digital input/output pins)   + Microcontroller to Cloud: MQTT (Adafruit IO broker service) * Implementation approach:   + The ESP32 connects to Wi-Fi and publishes sensor data using MQTT every 30 seconds or on state change.   + Subscriptions allow remote pump override from dashboard.   **Data Processing Method:**   * Cloud platform: [Adafruit IO Dashboard](https://io.adafruit.com/hcshah26/dashboards/hydroponic-project)   + Transmission method:     - Sensor data and pump status are published to the Adafruit IO feeds.     - Dashboard widgets display:     - Pump inflow and return flow rates (numeric L/min)     - Sensor alerts (toggle indicators)     - Pump control switch (bi-directional control)   + Visualisation:     - Real-time flow rates     - Digital indicators for overflow and warnings     - Feed values updated with each trigger or scheduled refresh * Local processing: ESP32   + GPIO pins to get inputs from flowmeters, float switch sensors and pump override switch   + LED lights to indicate states for flowmeters, float switches and pump override   + No local storage, all data is published to Adafruit IO via feeds.   **Power Management Considerations:**   * Power requirements:   + Simulated in Wokwi using constant USB power supply   + In real deployment, ESP32 and sensors would require 3.3 to 5V input and 250mA current. * Efficiency measures:   + Wi-Fi connection and MQTT updates optimised to only send data on state change   + Future enhancements could include sleep modes for battery powered systems   **Regulatory Compliance:**   * Applicable standards:   + Electrical Safety: Compliance with Australia/New Zealand standards (AS/NZS 3000)   + Data Privacy: Adafruit IO adheres to secure connection protocols and user credentials   + EMC: Certified microcontrollers and sensor components to reduce interference * Compliance measures:   + Use of published secure MQTT protocol   + Enclosure design and electrical insulation to prevent short circuits and hazards   **System Benefits:**   |  |  | | --- | --- | | Feature | Benefit | | Flow monitoring | Prevents water waste and detects flow anomalies | | Float switches | Warns of low water or overflow before damage occurs | | Remote dashboard | Allows monitoring and pump control from anywhere | | Alerting system | Provides proactive notification for intervention | | Simulation support | Enables safety testing without live water |  * Alignment with user stories:   + Maintenance technician receives alerts for water level issues   + System admin can remotely stop or start the pump and monitor usage trends   **Error Handling Approach:**   * Detection methods:   + Software debouncing for mechanical switch errors   + MQTT connection loss detection and reconnection logic   + Interrupt fallbacks to ensure pulse counts aren’t missed * Response procedures:   + If overflow or critical condition detected, pump is shutdown automatically   + Failed MQTT publishes are logged to serial output and retried   + Float switch and flowmeter toggles retrigger alert if stuck   **Key Features Summary:**  The hydroponic monitoring system developed in this project offers a robust and scalable solution for managing water flow and reservoir safety. Key features include:   * **Real-time Monitoring:** The system continuously tracks the inflow and return flow rates using simulated flowmeters, providing accurate readings in litres per minute. * **Cloud-Based Dashboard:** Integration with Adafruit IO enables remote monitoring and control, offering a user-friendly interface for viewing sensor data and managing system operations. * **Automated Safety Controls:** Built-in logic disables the pump automatically when critical conditions are detected – such as very low reservoir levels or pipe overflow – helping to prevent equipment damage or water loss. * **Responsive Sensor Handling:** Float switches are processed through hardware interrupts with software debouncing to ensure immediate and reliable state changes. * **Efficient Data Communication:** MQTT protocol is used to transmit and receive sensor data and control commands, minimising bandwidth use while maintaining real-time responsiveness. * **Manual Override Capability:** A local pump override switch and remote toggle via the dashboard allow users to intervene as needed for maintenance and emergency actions. * **Error Handling and Recovery:** The system includes mechanism to detect sensor faults and reconnect to the cloud platform if the MQTT connection is lost, ensuring resilience and uptime.   These features collectively support the system’s goals of improving efficiency, minimising waste and enhancing the reliability of the hydroponic actions.  **Task 4: Prototype**  **Prototype Type:**   * [ ] Hardware-based * [✓ ] Simulation – Using Wokwi ESP32-S2 board with virtual components * [ ] Hybrid   **Sensor Implementation:**   |  |  |  | | --- | --- | --- | | Sensor | Type | Function | | Flowmeter 1 (Inflow) | Simulated digital pulse | Measures out going water from the reservoir to hydroponic trough | | Flowmeter 2 (Return) | Simulated digital pulse | Measure returning water to the reservoir from the hydroponic trough | | Horizontal Float Switch 1 | Simulated digital | Detects low water warning level in the reservoir | | Horizontal Float Switch 2 | Simulated digital | Detects critical low water warning level in the reservoir and triggers pump shutdown | | Vertical Float Switch 1 | Simulated digital | Detect hydroponic trough overflow and triggers pump shutdown | | Pump Override Switch | Simulated digital | Allow manual toggling of the pump for maintenance or override purposes. |   **Data Transmission Method:**   * Protocol used: MQTT * Transmission frequency: Flowmeter data every 30s or immediately upon sensor state change * Error handling approach: * Debouncing Handling: Software debounce logic for all switches to prevent false triggers * MQTT Connection Loss: Automatic reconnection attempts on failure * Pulse Integrity: Use of hardware interrupts to ensure no loss of flowmeter pulses * Pump Auto-Shutdown: Activated immediately if overflow or critical reservoir level detected   **Testing Performed:**   * [✓] Normal operation test case * [✓] Error condition test cases (sensor failure, network disconnection) * [✓] Performance testing under load * [X] Battery life testing (if applicable) – Not applicable (simulation powered externally)   **Documentation Included:**   * [✓] Hardware schematic and wiring diagram * [✓] Software architecture diagram * [✓] Installation and setup guide * **GitHub Repository URL:** [**https://github.com/HCShah26/Capstone-IoT-Hydroponic.git**](https://github.com/HCShah26/Capstone-IoT-Hydroponic.git) * **Wokwi:** [**https://wokwi.com/projects/433543810637673473**](https://wokwi.com/projects/433543810637673473) * **Adafruit Dashboard:** [**https://io.adafruit.com/hcshah26/dashboards/hydroponic-project**](https://io.adafruit.com/hcshah26/dashboards/hydroponic-project)   **Bill of Materials:**  See detailed BOM for all component on the virtual simulation – Bill of Material.docx  BOM for Hybrid simulation (All component to build a physical simulator)   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Component | Qty | Description | Cost | Source | | ESP32 C3 Wroom | 1 | Main controller board | 26.80 | Core Electronics | | Flowmeter | 2 | Use of potentiometer | 0 |  | | Pump Override Switch | 1 | Use of Push button | 0 |  | | Horizontal Float Switch | 2 | Use of Push button | 0 |  | | Vertical Float Switch | 1 | Use of Push button | 0 |  | | LED | 100pk | For indicating state of sensors and pump override switch | 3.25 | Core Electonics | | Resistors | 4 | For LEDs | 0.40 | Core Electronics | | Push Button | 4 | Sensors and override switches | 3.80 | Core Electronics | | Potentiometers Breadboard Trim Pot 10k | 2 | For controlling the flowrate of the flowmeters | 5.70 | Core Electronics | | Breadboard ZY-102 | 1 | Circuit assembly | 4.60 | Core Electronics | | Jumper Wire Kit assorted 140pcs | 1 | Circuit assembly | 7.95 | Core Electronics | | Power supply | 1 | External 5V power supply | 12.05 | Core Electronics |   Estimated total: $64.55  BOM for actual sensors   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Component | Qty | Description | Cost | Source | | Flowmeter | 2 | Hall effect water flow meter YF-S201 | 11 | Altronics | | Pump Override Switch | 1 | Smart Plug | 26 | Altronics | | Horizontal Float Switch | 2 | Horizontal Tilt Float Switch | 40 | Altronics | | Vertical Float Switch | 1 | Vertical Tilt Float Switch | 18 | Altronics |   Estimated cost for actual sensors: $95  **Task 5: IoT Dashboard**  **Required Visualizations:**   * Real-time data display method:   + The dashboard charts the flow rate as litres per minute and status indicator for float switch sensors and override plug     - Pump inflow rate (pump-in-flow-rate)     - Pump return flow rate (pump-return-flow-rate)     - Reservoir Warning Indicator (reservoir-warning-alert)     - Reservoir Critical Indicator (reservoir-critical-alert)     - Tray overflow Indicator (pipe-overflow-warning)     - Pump Running (pump-override)   + Data updates every 30 seconds or upon system trigger   + LED indicators simulate current pump state and sensor status * Historical data visualization (hourly, daily, weekly):   + Adafruit IO’s line chart blocks (if added) can track:     - Flow rates over time     - Trigger frequency of warning / critical / overflow states   + This helps in analysing pump performance trends and water usage * System status indicators:   + Toggle indicators show:     - “pump-override”: ON/OFF state of the pump (controlled manually or automatically)     - “reservoir-warning-alert”: Horizontal Float Switch low water level warning status     - “reservoir-critical-alert”: Horizontal Float Switch very low water level Critical alert     - “pipe-overflow-warning”: Vertical Float Switch overflow trigger alert   These act as visual alarms and status feedback tools.  **Alert System Implementation:**   * Threshold-based alerts:   + Alerts are triggered automatically when:     - * Critical float switch or pipe overflow is HIGH       * System logic immediately disables the pump and updates the dashboard   + These alerts reflect physical dangers (empty reservoir, overflow) * Notification methods (email, SMS, push notifications):   + Adafruit IO supports:     - Email notification     - IFTTT integration for SMS or push alerts     - Future enhancements can integrate webhook actions for SMS/Email alerts via Zapier or IFTTT * Alert history tracking:   + Feed history on Adafruit IO stores all state changes   + Users can manually inspect logs or create charts for alert frequency   **Dashboard Platform:**   * Platform used: Adafruit IO * Configuration details:   + Feeds created:     - “pump-in-flow-rate”, “pump-return-flow-rate”     - “reservoir-warning-alert”, “reservoir-critical-alert”, “pipe-overflow-warning”     - “pump-override”   + Dashboard blocks:     - 2 Numeric display blocks (for flow meter rates)     - 4 Toggle blocks for sensors and pump control   + MQTT authentication via Adafruit username and AIO Key * Accessibility features:   + Web-based and mobile responsive   + Accessible from any device with internet access   + Simple readable labels and color-coded blocks   + Real-time feedback and two way control from any location   **Student Declaration**  I declare that this submission is my own work, and I have not plagiarised any content. All sources have been properly referenced and acknowledged.  **Signature:** \_\_\_\_Hiten\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Date:** \_\_30/06/2025\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Submission Checklist**   * [✓] Completed all required tasks (1-5) * [✓] Created a working prototype (hardware or simulated) * [✓] Set up a functional dashboard * [✓] Written comprehensive documentation * [✓] Created a public GitHub repository with all code * [✓] Prepared report or presentation slides * [✓] Included hard copies of all presentation materials/transcript * [✓] Included video demonstration of working prototype (5 minutes maximum) * [✓] Filled out this submission template completely |
|  |