UNIVERSITY OF PORTSMOUTH
FACULTY OF TECHNOLOGY

Academic Year 2024/25

M21410 Real-Time Embedded Systems

Coursework

Deadline For Submission: 21st May 2025 by 12:00 pm (UK time)

Hand-in Instructions: The submission process requires **two main documents** to be uploaded to Moodle:

1. A **PDF report** detailing your design and test results.
2. A **plain text** document with the Arduino code used for the practical tasks. This is the implementation of the design section and the source of the test results in the report.

You have the option to submit your code in the form of an Arduino project file; however, a simple plain text file is sufficient. This ensures that the work is easily accessible and reviewable.

Instructions for completing the assessment: For your submission, please adhere to the following guidelines:

1. The report must be submitted in PDF format, **not** as a Microsoft Word document.
2. Both the report and the code file - should be **named using your student ID** (for example, UP1234567.pdf).
3. There is a word limit of **4000 words** for the report, which does not include code. The page limitation is **20 pages**. Content exceeding the limit will **NOT be marked**.
4. Ensure that your student ID number is included on each page of your report; placing it in the header or footer is recommended for consistency.

Additionally, be mindful of the university's stance on plagiarism. The submission of work that is not your own or heavily based on someone else's work without appropriate citation is considered plagiarism and is subject to university sanctions. It's important to ensure that your work is original and properly cites any sources used.

Examiners: Dr Hongjie Ma

1 Introduction:

In this individual project, you are tasked with creating a control system for a dual-stage smart conveyor system using the Adafruit Feather board or online Wokwi ESP32 simulator, leveraging the capabilities of FreeRTOS and the Arduino development environment. The control system will manage sequential quality inspections, with length measurement and barcode scanning on the first conveyor, followed by weight verification on the second conveyor. The system must maintain real-time product tracking and data reporting while ensuring proper multi-core task coordination and system safety.

The primary objective is to filter out defective products based on their physical attributes and track quality metrics throughout the manufacturing process. While C is the recommended programming language, you may also use C++, challenging you to apply your engineering skills to solve real-world problems in embedded system design.

2 System Description

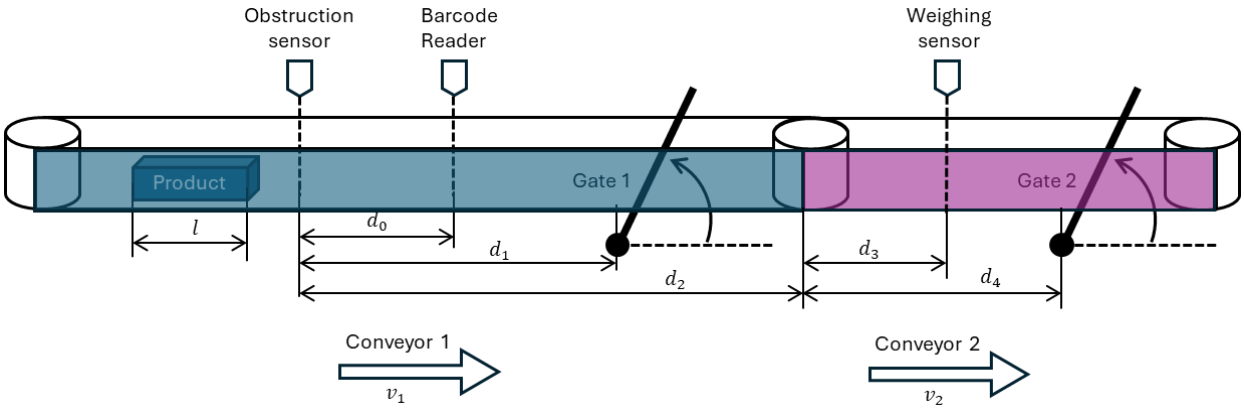

Fig.1 Conveyor system

As illustrated in Figure 1, the system consists of two cascading conveyor belts operating at different speeds to perform comprehensive quality control of products. Conveyor 1 operates at a constant speed of $v_1 = 1\ m/s$, while Conveyor 2 runs at $v_2 = 0.8\ m/s$. Due to the slower speed of Conveyor 2, there is a potential risk of product collision that must be managed by the control system.

2.1 Inspection Points and Measurements:

Stage 1 (Conveyor 1):

The first conveyor implements three key components for initial quality control:

1. An obstruction sensor detects, shown in Figure 1, product presence and enables length measurement through duration monitoring. Products with lengths ranging from $4.8\ cm$ to $5.2\ cm$ (inclusive) are considered acceptable.
2. A barcode reader is positioned $d_0 = 0.2\ m$ after the obstruction sensor. For accurate scanning, products must be precisely centred (±10% tolerance) beneath the reader when the read command is issued. Any misalignment or incorrect timing will result in failed scans.
3. Gate 1 is located $d_1 = 0.4\ m$ after the barcode reader (the distance between the obstruction sensor to the end of the conveyor1 $d_2 = 0.6\ m$). Products failing length specifications are diverted by this gate.

Stage 2 (Conveyor 2):

The second conveyor continues quality assessment with:

1. A weighing sensor positioned $d_3 = 0.2\ m$ from the start of Conveyor 2. Similar to the barcode reader, products must be centred (±10% tolerance) on the sensor when the read command is initiated to ensure accurate weight measurement. Products with weights ranging from 95 grams 105 grams (inclusive) are considered acceptable.

2. Gate 2 is located $d_4 = 0.4$ m from the start of Conveyor 2, providing final product sorting based on combined quality metrics.

2.2 Critical System Parameters

As shown in Figure 1, the dual-stage conveyor system implements multiple quality control checkpoints with specific timing and safety requirements. The following parameters are critical for the correct operation of the system show in Figure 1:

1) **Gate Response Time:**
   - Both gates (Gate 1 and Gate 2) have a mechanical response time of 2 ms
   - Gates must complete their position changes before products arrive
   - Control systems must account for this delay in timing calculations

2) **Safety Considerations:**
   - Due to v_2 being slower than v_1, product collisions on Conveyor 2 are possible
   - Any detected collision on Conveyor 2 requires immediate system emergency stop
   - Alarm must be triggered upon collision detection
   - System must maintain accurate tracking of all products in transit

3) **Timing Requirements:**
   - All sensor readings must be taken with products precisely centred
   - Gate positioning must account for the mechanical delay
   - Control system must coordinate timing between both conveyors to prevent collisions

4) **Quality Control Flow:**
   Products must pass through the system in sequence, with each inspection point contributing to the final quality determination. Failed products are diverted at their respective gates, while passing products continue through the complete system.

3 Controller Functionality Requirements

Your controller must implement the following functions across both conveyor stages, utilising multi-core processing for efficient real-time control and ensuring consistent data reporting formats:

1) Core Distribution and Task Management:
    a. **Core 1 manages Conveyor 1** control, length measurement, and barcode scanning
    b. **Core 2 handles Conveyor 2** control, weight measurement, and quality correlation
    c. Inter-core communication via shared memory for product tracking
2) Real-time Status Updates: The controller must automatically upload real-time status every second in the following format:
   "*convRT: AcceptCount,RejectCount,SN,Length,Weight,SN,Length,Weight,...*"
   For example: "*convRT: 082,008,1234,5.1,150.2,1235,4.7,140.2*"
   Where:
    - "*convRT*:" is the fixed header
    - *082* is the total count of acceptable products
    - *008* is the total count of rejected products
    - Following are the measurements for each product processed in that second:
        1) *1234* is the barcode number of the first product
        2) *5.1* is the length in cm
        3) *150.2* is the weight in g
        4) *1235* is the barcode of the second product
        5) *4.7* is the length in cm (rejected as out of spec)
        6) *140.2* is the weight in g

   All numeric values for Real-time Status Updates should be rounded to one decimal place for measurements. Product counts should be formatted as three digits with leading zeros. If a product is rejected at Gate 1 due to length specifications, its weight measurement will be recorded as 0.0 since it never reaches the weighing sensor on Conveyor 2.
3) User Query Response:
   The controller must respond to specific commands:
   a) command "stat": Upload statistical results
      Response Format: "*convSTAT: 5.08,5.02,5.42,152.1,150.2,155.8*"
      Where:
    - "*convSTAT*:" is the fixed header
    - *5.08* is the average total length
    - *5.02* is the average acceptable length
    - *5.42* is the average defective length
    - *152.1* is the average total weight
    - *150.2* is the average acceptable weight
    - *155.8* is the average defective weight

      All measurements for User Query Response rounded to two decimal places
   b) "reset": Reset all states, memory, and counters (no response required)
   c) "emergency": Trigger immediate system halt with status message
   Format: "EMERGENCY: Conveyor halted"

4) Quality Control Requirements:
    a) Measure and verify product length (4.8-5.2 cm acceptable range)
    b) Measure and verify product weight (95-105 g acceptable range)
    c) Scan barcodes when products are centred (±10% tolerance)

d) Measure weight on Conveyor 2 when centred
e) Maintain correlated quality data between conveyors

5) System Safety and Error Handling:
   a. Monitor speed differential between conveyors ($v_1 = 1\ m/s$, $v_2 = 0.8\ m/s$)
   b. Account for 2ms gate response time
   c. Implement collision detection on Conveyor 2
   d. Manage sensor failures and reading errors
   e. Ensure proper memory management and resource cleanup
6) All numeric outputs should be rounded to two decimal places unless specified otherwise. The system must maintain real-time performance while ensuring thread-safe operations between cores and consistent data reporting formats for upstream processing.

4 Implementation Guidelines and Requirements

4.1 Development Environment and Tools:

For this coursework, you may use either the online Wokwi simulator or the Adafruit Feather board provided during lab sessions. The serial port baud rate must be configured at ***115200bps*** for all communications.

A Conveyor simulator library and example code will be provided through Moodle. This simulator facilitates development by providing sensor readings, gate status monitoring, and ground truth data for product lengths and weights to support grading. You are encouraged to develop additional simulation tools to support your development process.

4.2 Academic Integrity:

This is an individual project that requires original work. Code sharing between students is strictly prohibited and will result in mark penalties. All submissions undergo thorough manual review by assessors. Attempts to conceal plagiarism through function or variable name modifications will be detected and penalised.

4.3 Data Format Requirements:

The system's output formats for real-time status updates and user query responses must strictly adhere to the specified formats **described above**, as automated assessment tools will process these outputs. Any deviation from the required formats will result in mark penalties.

4.4 Code Quality and Assessment:

Meeting all specified functionality requirements does not automatically guarantee full marks. The assessment considers additional factors including:

- Code reliability and robustness
- Presence of potential bugs or edge cases
- Quality of implementation
- System performance

4.5 Implementation Requirements:

1) Ensure clear separation between real-time status information from both conveyor belts and user query messages.
2) When using the template code provided (excluding simulator components):
    - Usage is permitted but must be thoroughly documented in your report
    - Partial marks will be awarded for sections using template code
    - Your report must clearly explain your understanding and any modifications

The final assessment will evaluate both the completeness of implemented features and the quality of their implementation, with emphasis on reliable real-time performance and robust error handling.

5 Marking scheme

1) Refer to the Coursework's cover page for submission guidelines.
2) Use the provided template for formatting your report. This template is available for download from this link (**click here**).
3) The coursework is worth a total of 100 marks, divided equally between code (50 marks) and a report (50 marks).

5.1 Code Evaluation:

Ensure your submitted code is executable within the described software and hardware environment. Failure to execute may result in a lower score as the assessor cannot accurately evaluate the implemented functionalities.

Functionality evaluations and their corresponding marks (totalling 50 marks) are as follows:

Assessment Criteria	1-2 marks	3-4 marks	5-6 marks	7-8 marks	9-10 marks
System Implementation *(Up to 10 marks)*	Basic implementation with significant errors	Functional core distribution and basic memory management	Proper task management and memory handling, basic safety features	Efficient core utilisation, verified memory management	Optimal performance with profiling evidence, comprehensive safety features
Real-time Control *(Up to 10 marks)*	Basic sensor reading and control implementation	Functional control with occasional timing issues	Consistent timing control handles normal operations	Robust handling of sensor/actuator faults	Comprehensive fault handling, <50ms emergency response
Data Processing *(Up to 10 marks)*	Basic data collection with errors	Reliable data processing with occasional misreads	Accurate measurements, proper tracking	100% barcode read accuracy, correct weight measurements	Perfect handling even with sensor/actuator faults
Communication Implementation *(Up to 10 marks)*	Basic communication with format errors	Functional communication, occasional delays	Correct format, stable communication	Zero format errors, handles abnormal inputs	<100ms response time, zero transmission errors
Code Quality and Documentation *(Up to 10 marks)*	Poor code organisation, minimal documentation	Basic structure and error handling	Clear layout, functional error recovery	Professional organisation, comprehensive error handling	Exceptional documentation, complete resource management

5.2 Report Evaluation

Assessment Criteria	1-2 marks	3-4 marks	5-6 marks	7-8 marks	9-10 marks
System Design and Architecture *(Up to 10 marks)*	Basic system overview with incomplete flowcharts. Poor illustration of system structure. No clear explanation of multi-core distribution. Less than 2 system diagrams provided.	Limited system documentation with basic flowcharts. Basic data flow explanation. Minimal discussion of task distribution. 2-3 relevant diagrams included.	Adequate system documentation with clear flowcharts. Satisfactory explanation of data flow and task distribution. 3-4 relevant diagrams with explanations.	Comprehensive system documentation with detailed flowcharts. Clear illustration of data flow and task distribution. 4-5 well-documented diagrams with detailed explanations.	Exceptional system documentation with professional flowcharts. Detailed analysis of data flow and task distribution. >5 high-quality diagrams with comprehensive explanations. All design decisions thoroughly justified.
Technical Implementation *(Up to 10 marks)*	Basic description of system components. Minimal documentation of task definitions. No clear explanation of resource management.	Limited documentation of system components. Basic coverage of task interactions. Simple explanation of resource management strategies.	Clear documentation of system components and task definitions. Adequate coverage of resource management and critical sections. Basic technical rationale provided.	Detailed documentation of all system components. Comprehensive coverage of resource management strategies. Clear technical rationale for design decisions.	Exceptional documentation of all system aspects. In-depth analysis of resource management strategies. Comprehensive technical rationale with performance considerations.
Testing and Validation *(Up to 10 marks)*	Basic test strategy with minimal test cases. No clear test objectives. Limited coverage of system functionality.	Limited test strategy with few test cases. Basic test objectives stated. Some coverage of normal operations.	Clear test strategy with adequate test cases. Defined test objectives. Coverage of normal operations and some edge cases.	Comprehensive test strategy with detailed test cases. Clear objectives and expected outcomes. Coverage of normal operations, edge cases, and error conditions.	Exceptional test strategy with extensive test cases. Detailed objectives, outcomes, and results. Complete coverage of all scenarios including system recovery.
Document Layout and Structure *(Up to 10 marks)*	Poor adherence to template. Inconsistent formatting. Limited use of technical illustrations. Multiple language/grammar issues.	Basic template adherence. Inconsistent formatting in places. Basic technical illustrations. Some language/grammar issues.	Good template adherence. Mostly consistent formatting. Adequate technical illustrations. Minor language/grammar issues.	Strong template adherence. Consistent formatting throughout. Effective technical illustrations. Clear technical writing.	Excellent template adherence. Professional formatting throughout. High-quality technical illustrations. Outstanding technical writing.
Critical Analysis and Reflection *(Up to 10 marks)*	Basic system evaluation. Few limitations identified. No clear improvement suggestions. Limited understanding shown.	Limited system evaluation. Some limitations identified. Basic improvement suggestions. Some understanding shown.	Clear system evaluation. Key limitations identified. Reasonable improvement suggestions. Good understanding demonstrated.	Comprehensive system evaluation. Most limitations identified. Well-thought-out improvements suggested. Strong understanding demonstrated.	Exceptional system evaluation. All limitations addressed. Innovative improvements suggested. Excellent understanding of broader implications.

Notes:

1. System diagrams must include clear labels and explanations
2. Design decisions must be supported by technical rationale
3. Testing documentation must include specific, measurable outcomes