CENTRE FOR PRODUCT DESIGN &
MANUFACTURING
Indian Institute of Science

Embedded systems task

Table of contents

Background

Embedded systems role @CPDM, IISc

The role will consist of the following responsibilities:

- Developing a scalable framework for firmware architecture

- Working with and interfacing sensors for temperature, weight, analog sensors.

- Develop low-level drivers for communicating with an onboard processor.

- Writing firmware for an ARM Cortex-M4 based microcontroller for managing all hardware resources.

- Development of GUI using python on a Linux based system.

Task description

This task requires you to write firmware drivers and code in Embedded C for achieving the following objectives:

1. The user sends a number between **0 to 100** from the PC (Laptop, Desktop) to the MCU.
2. A UART **communication driver** in the MCU is responsible for receiving this data.
3. The above driver also needs to conduct a **CRC-16 data integrity check** to ensure that the data is not corrupted.
4. This driver will also **send a confirmation back to the PC** that the data received is correct.
5. The driver pushes this **data directly into memory** through Direct Memory Access (DMA)
6. Once this data is in memory, two things need to happen:
    a. The On-Board test LED is blinked by PWM for a total period of 10 seconds at a duty cycle equal to the number received from the User.
    b. If the number is a **multiple of 4,** the output needs to be sent to the PC which is the string "CPDM"; if the number is a **multiple of 7,** the output needs to be sent to the PC which is the string "IISc"; if the number is a **multiple of both 4&7**, then an output needs to be sent to the PC which is the string "CPDM IISc"; if the number is neither a multiple of 4 or 7, output to PC is the number itself. **A data integrity check for sending data back to the PC is also necessary**.
7. The system should be dynamic such that if another input is received from the user within 10 secs from the last input, **it is pushed into a queue**. The output is sent to the PC and the LED starts blinking at the new rate only after the 10 seconds for the previous input are completed.
8. Corresponding code for sending data from the PC to the MCU also needs to be written. This should be written using Python and an executable basic GUI.

Examples

User Input: 56(@Start: 0 sec), 21(@4 sec), 87(@42 sec)
LED Output: [Blinking at 56% (b/w 0-10sec), Blinking at 21%(10-20sec),
LED OFF (20-42sec), Blinking at 87%(42-52sec)]
PC Output: [CPDM IISc (at 0th second), IISc (at 10th second), 87 (at 42nd second)]

MCU

The above code needs to be written in Embedded C for a microcontroller with an ARM Cortex-M4 core. Some examples are:

1. STM32F4 Series or STM32L4 series (Recommended)
2. LPC1768 from NXP

If there is another MCU you might want to use, you are free to. If you have the actual hardware with you, nothing like it!

We know that debugging without the actual MCU hardware board will be difficult, but we'll be judging you based on:

1. Please think carefully about the structure of the queue, is it a ring buffer or a FIFO queue or something else?
2. Give thought to the logic structure of finding the multiple of 4,7
3. Overall structure and flow of the code logic.
4. Also, think, is this code framework scalable? Can it be extended to other use cases if some basic conditions like output string, multiplier number change?
5. Once written, please think how easy this will be to maintain and debug for a person you don't know.

What we need:

Please compress the following files into a **single ZIP file**, upload the file to a cloud drive (Google drive/Dropbox etc.) and **share the link as a message on Indeed**. Please make sure you **change the permissions** on the link so that it is accessible to everyone.

- A brief PDF explaining the code along with the Readme.
- Project files.
- Compile instructions which should work without any problems.
- Properly citing all the open-source frameworks/references and GitHub codes if used.

What will you be judged on:

- Scalability
- Performance and benchmark
- Code quality