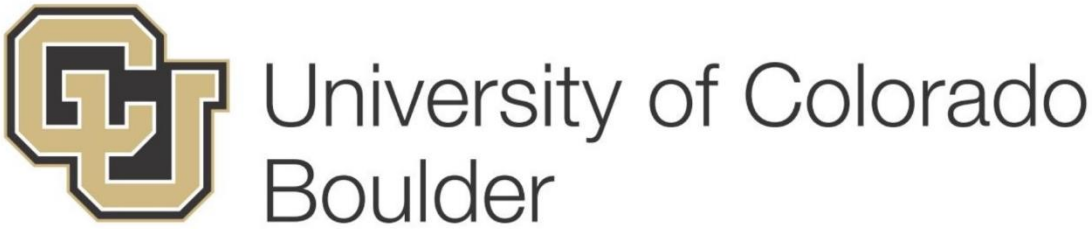


ADVANCED EMBEDDED SYSTEM DEVELOPMENT

PROJECT 2–

SYSTEM & ARCHITECTURE DOCUMENT

Deepesh Mahendra Sonigra

Madhumitha Tolakanahalli Pradeep

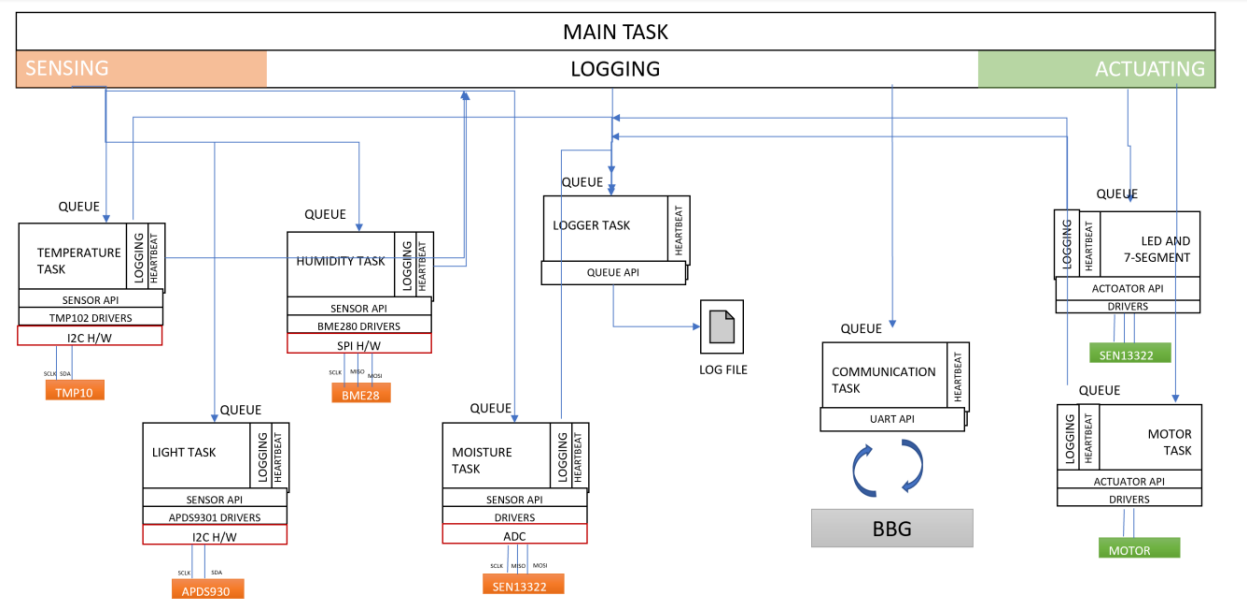
Github Repository: https://github.com/deep6000/Smart-Plant_Irrigation-System.git

Project Requirements

1) The Remote node senses the humidity in air, moisture in soil, temperature and light during day as well as night and communicates it to control Node. Control Node takes decision based on the values and communicates to remote node the amount of water to be irrigated.
2) If any of the sensors fail, the system based on the value of other sensors approximates the results and works in degradable mode.
3) The system should perform a self in build test on startup and on failure of any sensor(s) the system retries for a timespan. If built in self-test still fails, based on the results, the system continues to work in degradable mode
4) The system displays temperature moisture or humidity on 7 segment display based on the user requirements.
5) The buzzer is activated if moisture in soil rises the threshold value indicating fault condition.

Architecture Description

Remote Node



Application Tasks

> Main Task

The Main Task is the parent thread, implemented using asynchronous threading strategy to manage child threads. The Main Task performs following functions -

- Child Thread Management Function
- Controlling Actuators

The main task creates threads that perform the following functions –

- Temperature Sensor Task
- Light Sensor Task
- Moisture Sensor Task
- Humidity Sensor Task
- Synchronized Logger Task
- Communication Task

CHILD THREAD MANAGEMENT FUNCTION

> The parent thread keeps the track of children tasks to check if they are alive and running on specified time intervals or dead and stuck for a specific timeout.

> This can be achieved by using Heartbeat notification from all child threads to the parent thread using request response mechanism.

> When requested to stop the task, the main task should clean up everything, issue exit commands to the child threads and terminate gracefully

CONTROLLING ACTUATORS

> Based on the user requirements display values of temperature, humidity and moisture

> Based on the values of sensors turn on/off the motor

> Turn on buzzer on identification of fault conditions

TEMPERATURE TASK

> This task is concerned with the Temperature Sensor TMP102 with I2C. It wakes up at a regular interval to collect data from the sensor.

> It consists of read and write functions for sensor registers and modify configuration registers.

> API calls

getTemperature()

LIGHT SENSOR TASK

> This task is concerned with the Light Sensor APDS-9301 Light Sensor. It wakes up at regular intervals to collect data from the sensor.

> Sample API List

APDS9301_GetState(uint * LUX); //Get state light/dark
APDS9301_OutputLuminence(int * temp);

MOISTURE TASK

> This task is concerned with the moisture Sensor SEN133322 Sensor. It wakes up at regular intervals to collect data from the sensor.

> Sample API list

GetMoisture();

HUMIDITY TASK

> This task is concerned with the humidity Sensor BME280 Sensor. It wakes up at regular intervals to collect data from the sensor.

> Sample API list

Gethumidity();

SYNCHRONIZED LOGGER TASK

> This task is concerned with logging data from various on-board sources. As this interface is going to be accessed by multiple sources, synchronization is needed to maintain data consistency. In this project, data protection is achieved using **Message Queue.**

> The log file path and file name are configurable. This is accomplished using command line options.

> Once the process is requested/forced to close, it should close all file handles and terminate gracefully.

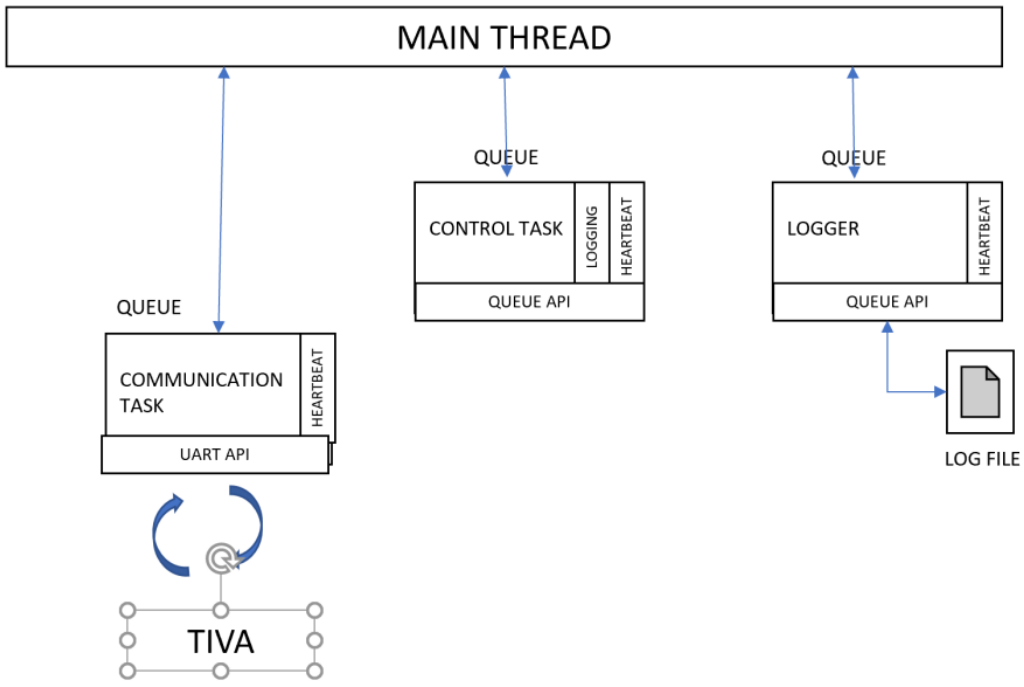
> Each log entry contains the following information: timestamp, log level, logger source ID, Log Message

COMMUNICATION TASK

> This task acts as an interface to Remote Task and Control Task

> The Remote Task and Control Task communicates using NRF protocol

CONTROL NODE



Application Tasks

> Main Task

The Main Task is the parent thread, implemented using asynchronous threading strategy to manage child threads. The Main Task performs following functions -

- Child Thread Management Function

The main task creates threads that perform the following functions –

- Logger Task
- Communication Task
- Control Task

LOGGER TASK

> This task is concerned with logging data from various on-board sources. As this interface is going to be accessed by multiple sources, synchronization is needed to maintain data consistency. In this project, data protection is achieved using **Message Queue.**

> The log file path and file name are configurable. This is accomplished using command line options.

> Once the process is requested/forced to close, it should close all file handles and terminate gracefully.

> Each log entry contains the following information: timestamp, log level, logger source ID, Log Message

COMMUNICATION TASK

> This task acts as an interface to Remote Task and Control Task

> The Remote Task and Control Task communicates using NRF protocol.

CONTROL TASK

> The Control task has an algorithm based on the sensor values decides the outcome, that is, the amount of water to be irrigated.

>

PROJECT PLAN

Task	Start Date	End Date	Days
Planning and Architecture Diagram	07-Apr	10-Apr	1
Interfacing Moisture Sensor	08-Apr	08-Apr	1
Interfacing Humidity Sensor	09-Apr	09-Apr	1
Interfacing Temperature Sensor and Light Sensor	10-Apr	10-Apr	1
Interfacing between BBG and TIVA	11-Apr	14-Apr	3
Control Algorithm on BBG	15-Apr	18-Apr	3
Sending back actuator data to TIVA	18-Apr	18-Apr	1
Interfacing 7Segment Display	19-Apr	19-Apr	1
Interfacing Motor	20-Apr	20-Apr	1
Implementing actuator functionalities	21-Apr	22-Apr	2
Extra credit functionalities	23-Apr	25-Apr	2
Integration	26-Apr	26-Apr	1
Testing and bug fixing	27-Apr	28-Apr	2