UFSC 2022 - Embedded Systems Programming in C++

Weather station Embedded System

Gabriel de Oliveira Aguiar[1]

[1]UFSC - Universidade Federal de Santa Catarina

November 24, 2022

Abstract

This project consist of the implementation of a Weather station that uses a machine learning algorithm to determine the next temperature and humidity. The main objective is to reduce energy waste and environmental impact.

The system is divided in three main components. First component is a embedded system inside of a Raspberry Pi, which is responsible for data acquisition and processing. Second is a host computer, responsible for storing a event log sent by the embedded system and displaying this information. Third is a host smartphone, with the same responsibilities as the host computer but using wireless to acquire and send data.

The three of them are developed using C++ with an object-oriented developing, designed to be easily adaptable to different weather station system manufacturers and operating systems.

Keywords
Embedded System, Weather Station, Raspberry, Efficiency, C++, Machine Learning

1 Introduction

A weather station is a facility, either on land or sea, with instruments and equipment for measuring atmospheric conditions to provide information for weather forecasts and to study the weather and climate.

The measurements taken include temperature, atmospheric pressure, humidity, wind speed, wind direction, and precipitation amounts. Wind measurements are taken with as few other obstructions as possible, while temperature and humidity measurements are kept free from direct solar radiation, or insolation. Manual observations are taken at least once daily, while automated measurements are taken at least once an hour.

Weather conditions out at sea are taken by ships and buoys, which measure slightly different meteorological quantities such as sea surface temperature (SST), wave height, and wave period. Drifting weather buoys outnumber their moored versions by a significant amount.

The important thing is that most of the measurements is the same (close) or predictable. We can therefore decrease the number of records using a machine learning model to predict most temperatures and only record outliers. Thus, reducing energy expenditure.

2 Proposed design, Materials & Methods

The system was implemented with a focus on Object-Oriented Programming and C++ language, using the Oracle VM Virtual Box to use Linux (Debian Distro) as development environment and a GitHub repository [1]for version control, project management and code sharing.

The development of this project is part of the course EEL7323 - Embedded Systems Programming in C++, the complete project requirements provided professor Eduardo Augusto Bezerra can be found in his website and in the project's GitHub repository's Documentation Folder.

The system is designed to learn how the weather beheaves using machine learning (abbreviated to ML from here on) to reduce energy consumption. The system evaluates environmental data and learns weather characteristics to determine which registerments to do.

The system is divided in three main components, the embedded system, the host computer, and the host smartphone.

The embedded system's macro functionalities are:

- Acquire environmental information through sensors;

- Pass this information as input to the ML algorithm;
- Generate logs registers based on the output of the ML algorithm and sensors;
- Generate a log of all events with detailed information;
- Send log information to host computer and smartphone when requested;

For the hardware will be used:

- Temperature sensor name;
- Humidity sensor name;
- Raspberry Pi 3

2.1 Host computer software

The host computer software was compiled using g++, and uses the GNU Make build system. The source code for the host computer software can be found at the project's GitHub repository[1].

The host computer software must accomplish the following tasks:

- Establish serial communication with the embedded system;
- Store the log data received from the embedded system in a queue;
- Display the available log information in two ways:
  -List all events between two dates.
  -Get data, sent or not, between two dates.

The software uses a terminal interface to interact with the user, providing a menu with two options, as mentioned above.

The embedded system interface was heavily based on the RobotLinux example provided in class. Using an abstract base class to provide an interface with the embedded system, and derived classes implementing the methods for different operating conditions, in this case, a derived class that supports serial communication in a Linux environment.

The embedded system interface class implements the main methods necessary for the requested program functionalities, with functions dedicated to opening and monitoring a serial port, methods for listing the log information from the embedded system, and a queue to store the log information.

The embedded system periodically sends the available log information through a serial port, with this in mind, the host software is designed

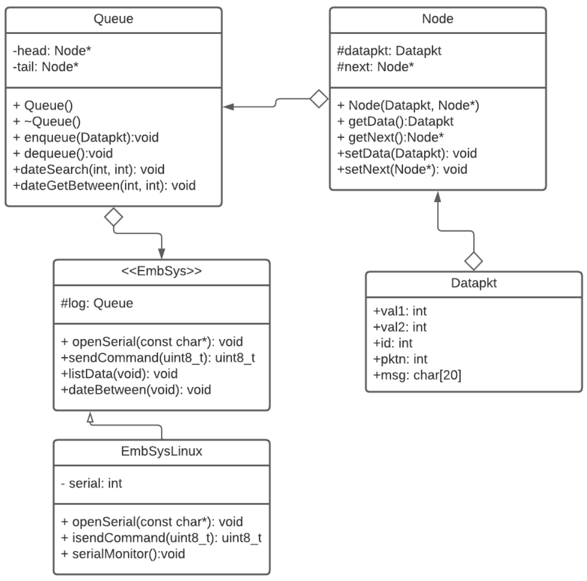

Figure 1: UML Diagram Computer Software

to constantly monitor the serial port waiting for data, through a thread. The serial monitoring method, designed to run on a separate thread, registers any incoming information in the log queue. The thread that monitors the serial port is started as soon as the program starts and notifies if the serial port was opened successfully.

There is also a method to send a command to the embedded system, which consists of a pattern of three of the same, specific, predefined character sent consecutively. This command simply triggers the transmission of available log information in the embedded system.

The queue and its node are both implemented as classes, related by aggregation, providing the basic functionality of a queue data structure.

The Queue class has methods for enqueueing and dequeueing nodes, searching and listing information, and a two node pointers as attributes, called "head" and "tail", to determine the beginning and end of the queue.

The Node class attributes are a node pointer to indicate the next node, and the data packet with the log information, with basic setters and getters methods.

The data packet itself is also a class, able to store the raw data received from serial communication, and providing methods to translate this raw data into readable information.

2.2 Embedded software

to do...

2.3 Smart phone software

to do...

2.4 Integration and test

to do...

3 Results

to do...

4 Discussion

to do...

Conclusions

to do...

Acknowledgements

to do...

References

[1] Gabriel de Oliveira Aguiar. Project's GitHub.