COMP4900AF22 - Project Proposal
Real-Time Agricultural Response System (RARS)

Ben Mask, Andrew Heron, Sebastian Navas Chaparro, Massimo Gillis

Oct. 2022

Dr. Ahmad Traboulsi

Video Link: https://youtu.be/etJNJASpIkU

Introduction

The following document will outline the project proposal for the Real-Time Agricultural Response System (RARS) developed on the QNX Neutrino platform. The goal of this application is to produce a multi-threaded system that monitors simulated sensor input pertaining to the relevant variables of an agricultural environment and prompts responses from components within the system that are responsible for managing various actuators of differing levels of priority.

Project Description

Domain

The Real-time Agricultural Response System (RARS) would be developed with a focus on applications in the domain of agriculture. Agriculture is a field which has a wide scope. While the domain of agriculture can include animal husbandry, cultivating soil conditions, and the growth of crops, the focus of the RARS application would primarily be on the latter of these three. The RARS would ideally be implemented in a small-scale, controlled, growing environment where it is feasible that actuators could reliably adjust the surrounding environment in response to sensor readings in the environment. One such example would be a greenhouse.

Problem Statement and Project Purpose

Awareness of the need for sustainable agricultural practices has increased in recent decades, in tandem with the technological capacity for remotely monitoring parameters relevant to crop health and production. As a result, there is a growing opportunity for the introduction of automated systems that maintain ideal and sustainable growing conditions in response to changes in the surrounding environment in real time. The purpose of the Real-Time Agricultural Response System (RARS) is to simulate one way in which a responsive system could leverage the QNX Neutrino platform to achieve this kind of system.

The predominant parameters affecting growing conditions are the proportions of various substances which make up the soil composition, such as pH, nitrogen, and moisture, as well as, the atmospheric conditions of light, humidity, and CO_2 quantities. Given that these variables are not isolated but often have relationships, the goal of the RARS is to maintain an ideal growing environment in response to changes in these environmental conditions.

In order to maintain proper environmental conditions, the RARS application will take periodic measurements of each identified variable and activate actuators as needed to adjust environmental conditions.

Associated Technologies and Frameworks

The RARS application will be an RTOS application which leverages the features of the QNX Neutrino real-time operating system. The system will be implemented in the C programming language.

The application aims to simulate changes in the environment as well as the functions of various hardware components such as temperature sensors and heaters, or coolers in the environment. By simulating the functionality of the hardware components, the team will limit the scope of technologies to those provided by QNX and the C language.

The implementation of the RARS application will apply various features provided by the QNX Neutrino real-time operating system. Real-time operating systems allow developers to ensure that the execution of threads and processes takes place in a timely and deterministic manner. In QNX neutrino this is possible through pre-emptive scheduling and the prioritization of tasks.

A distinct feature of the QNX Neutrino OS that the team aims to make use of in the architecture of the system is Neutrino's message-passing functionality. In Neutrino, message passing is a core concept that promotes the development of encapsulated, modular processes by providing a native means of IPC between processes. In our system, we intend to use this to provide a level of abstraction between components such that individual components can interact through well-defined interfaces while remaining agnostic to the underlying business logic of any other component.

The QNX Neutrino OS provides C libraries which allow for C applications to interface with the OS' Kernel via Kernel calls. The RARS application will be developed in C and work with these libraries to implement aspects of the application design. Some functionality provided by these libraries that the application will use include multithreading, multiprocessing, and message passing.

To further extend the functionality of the application, the team considered running the application with hardware on a Raspberry Pi. A Raspberry Pi is a small computer and microcontroller frequently used in prototyping hardware applications. While migrating from a simulated environment to hardware controlled by a Raspberry Pi is not currently the intention, adopting this technology could be a future extension of the project.

With respect to experience in the technologies discussed above, no team member has previously worked with the QNX Neutrino platform. Learning how to effectively design our system in a way that makes use of the Neutrino features in an effective and idiomatic way will be a process for the team throughout the development of the project. The C programming language is a language that each team member has used in an academic setting. This will provide the benefit to the team of having a significant degree of orientation with the implementation language as we proceed with the proposed project.

Application Design and Study Method

The fundamental goal of the RARS is to improve agricultural growth by reading data from sensors in an agricultural environment and using a set of actuators to modify this environment in order to create ideal conditions. A secondary goal would be to display and log data received from the sensors in real-time to provide farmers or other agricultural workers with metrics for a better understanding of their growing environments.

In order to achieve the first goal, the RARS application needs to be able to do three things; read data from a sensor, make a decision about what that reading means, and activate some kind of actuator for controlling the element of the environment in question. To increase the usefulness of this application, it is also desirable that the RARS be able to perform these operations with multiple pairs of sensors and actuators. In order to achieve the second goal of logging the observed data, the system must also be able to store readings and other useful information in an organized manner and display it to a user.

Currently, we are investigating two possible designs for this application. The first design involves a central control loop which receives data from an array of sensors and uses that data to make decisions about which actuators to engage. This approach would create easily readable code, and simplify the prioritization of tasks. The drawback of this approach is it creates a central point of failure and scalability problems that the QNX Neutrino design philosophy documentation warns against.

Alternatively, we were also looking at taking a more decentralized approach, involving multiple processes which each manage a very small (usually size 1) set of sensors coupled to a corresponding set of actuators. Each of these processes would only be responsible for taking readings from their sensors and making decisions involving the activation of their actuators. This approach is more scalable and robust in terms of having multiple processes which can independently continue even if there is a problem with other parts of the system. The drawback however is that it could potentially cause interference issues if multiple processes are needed to make decisions regarding a shared actuator. In the system, we are designing this is not an issue, but it does limit the extensibility of the application in that regard. Furthermore, in order to display information to a user in real-time, all of these processes would still need to communicate with a central service. This is less of a scalability problem than the first approach, as a display service would be less resource intensive and less critical to the overall function of the application than a centralized control loop, however, it does still create a single point of failure.

In terms of simulating the sensors and actuators used in this application, we plan on doing so with another process. For each sensor present in the application, the process will write to a pipe a random value within a range. The sensor drivers will then read from their respective pipes in order to collect data for the application. If the application engages an actuator, the actuator driver will send a message to the simulation process, which will cause it to adjust the range of possible values for the corresponding pipe. We plan on simulating temperature, light, humidity, and PH sensors, as well as actuators for heaters, air conditioners, lamps, and a sprinkler system.