Milestone II Progress Evaluation

Project Title: Autonomous Multi-Cycle Farming in Space

From CSE [coordinator and project sponsor: Philip Chan, pkc@cs.fit.edu]:

Christopher Millsap <cmillsap2013@my.fit.edu>
Giampiero Corsbie <gcorsbie2018@my.fit.edu>

From BIO [coordinator: Drew Palmer, apalmer@fit.edu]:

David Masaitis <dmasaitis2016@my.fit.edu>

From MEE [coordinator: Elisabeth Kames <ekames2011@my.fit.edu>]:

Dominic Allard, dallard2016@my.fit.edu
Courtney Cline, ccline2018@my.fit.edu
Joseph Luya, jluya2016@my.fit.edu
Bryce Johnson, bjohnson2016@my.fit.edu,
Philip Bernhard, pbernhard2015@my.fit.edu,
Timothy Frazier, tfrazier2016@my.fit.edu,
Joshua Calhoun, calhounj2008@my.fit.edu,
Kali Jenson- AEE student, kjenson2015@my.fit.edu,
Chris Mateo- ECE student, cmateo2016@my.fit.edu

Clients:

Some potential clients: NASA, SpaceX, Blue Origin, Virgin Galactic.
Current client: Dr. Philip Chan Associate Professor, Computer Engineering and Sciences at Florida Institute of Technology, coordinator and project sponsor.

Progress Matrix

Task Matrix for Milestone 2

Task	Giampiero	Christopher	Completion	To Do
Create state diagram for the system	Responsible - 50%	Responsible - 50%	100%	none
Implement inheritable high-level interfaces/classes for sensor/actuator communication	Responsible - 50%	Responsible - 50%	100%	Branch Actuator into ActuatorSwitch and ActuatorSystem
Create simulated farm for testing	Responsible - 50% Test	Responsible - 50% Test/Demo		Create simulated farm for all phases
Create a configuration file for at least one phase in system and run in simulation	Responsible - 75% Test/Demo	Responsible - 25% Test/Demo	100%	Determine config file readability UX

Discussion

Task 1 - Create state diagram for the system:

To represent the states of the scheduler in a structured format for control flow so we can reference it during development. The idea behind this task was influenced by testing. We wanted to visualize the system in a way to remove potential bugs while drawing the diagram. Software development also flowed from this.

Task 2 - Implement inheritable high-level interfaces/classes for sensor/actuator communication:

For this task we wanted to create abstractions on the sensors and actuators that would be present in a real system. Activate method to initialize actuator parameters and to communicate with hardware, if it behaves like a switch. A deactivate method is also included. Contains a read function for powering on the sensor, reading in data from the hardware, then powering off the sensor. Has a frequency attribute, where “read” operations are rescheduled based on it’s value Implemented height sensor and moisture sensor classes for testing. Similar to sensor class, with actuators we implemented water pump and light classes for testing Classes contain activate method to initialize actuator parameters and to communicate with hardware, if it behaves like a switch.

Task 3 - Create simulated farm for testing:

The simulated farm was designed to be scalable with our system as we add more phases to the config file. Right now the simulator runs on the grow phase with classes that represent seeds that germinate and grow, moisture/water level in system that changes based on water pump, and functions added for future simulations. The actuator class in the simulator contains standard parameters for a motor.

Task 4 - Create a configuration file for at least one phase in system and run in simulation:

To command the system we needed a parsable set of configurations for the various expected system/crop behaviors. We agreed on using YAML and setup a parser defining reserved words, unit conversions, operation names, etc. Each phase consists of START, MONITOR, and END entries where the entry for MONITOR contains sensor names, whose entries are multiple ranges of values, each of which correspond to a sequence of config actions. START contains a sequence of config actions that initialize the system for a phase and END contains a sequence of config actions that are executed on transitions. Each config action comprises as a 3 part sequence, the first element is the operation “activate”, “set”, or “deactivate”, the second is the target of the operation which is either an actuator or sensor name. The third is parameters, each with a corresponding value. The configuration file parser:

- Converts the config file to the systems internal representation of the configuration.
- Validates the internal structure.
- Validates the syntax against reserved words and whether targets contain parameters specified.
- Normalizes units provided to a standard internal representation.
- e.g. minutes, hours, and days get converted into seconds.

Team Discussion

Notes are given based on interactions between CSE. They do not necessarily reflect the team member's entire contribution.

CSE:

- Christopher Millsap <cmillsap2013@my.fit.edu>:
    - Designed the simulation farm and integrated it into the system for testing and presentation
    - Found YAML through research
    - Helped create/design the Driver, Task Manager, high level classes for sensors/actuators and the configuration file
    - Prepared and finished the evaluation document
    - Prepared the simulation video
    - Presented demos
    - Helped create state diagram
- Giampiero Corsbie <gcorsbie2018@my.fit.edu>:
    - Provided python expertise
    - Main creator of the Driver, Task Manager, high level classes for sensors/actuators and the configuration file
    - Presented demos
    - Helped create state diagram

Plans for Next Milestone

Task	Giampiero	Christopher
Decide which set of computers/microcontrollers will be used to control the system other than the microcontroller	Responsible - 50%	Responsible - 50%
Implement Python libraries for serial communication	Responsible - 50%	Responsible - 50%
Create development toolchain (maybe do this next semester when arch is decided?)	Responsible - 50%	Responsible - 50%

Start a preliminary product breakdown structure with MEE	Responsible - 50%	Responsible - 50%
At least one sensor & actuator functioning according to some schedule ○ These will run based on simulated data along with simulated cycles	Responsible - 50%	Responsible - 50%
Full simulated farm with well-defined input parameters	Responsible - 50%	Responsible - 50%
Configuration file for all phases in system to run in simulation	Responsible - 50%	Responsible - 50%
Testing Document inheriting from test plan from milestone 1	Responsible - 50%	Responsible - 50%

Task 1 - Decide which set of computers/microcontrollers will be used to control the system other than the microcontroller:

We will need other smaller boards/hats to control peripherals and actuators on the system; this is most likely probable with the creation or integration of a robotic seeding mechanism and integrating canopy lighting. Currently the MEE team is looking to use a custom made mechanical apparatus with three actuators for moving in 3D space and seeding, for this we would need hats particular to that subsystem. For lighting we will need to monitor power consumption but not provide it through the board along with switching the states/frequency.

Task 2 - Implement Python libraries for serial communication

Controlling and synching subsystems will require more python libraries on the microcontroller. One of the system requirements states that ethernet will be the locker to system interface so we already know that this architecture may need serial libraries. In many cases, serial is cheaper to implement than parallel. Many ICs have serial interfaces, as opposed to parallel ones, so that they have fewer pins and are therefore less expensive.

Task 3 - Create development toolchain (maybe do this next semester when arch is decided?):

Good DevOps practices dictate that we have our tools used for the final product listed and linked. There is no one tool that will or has provided everything we need to develop the AMCFD. For this project we will have the support and library defined along with linking together programs used in dev by specific stages. During this milestone we will define the toolchain building process that best match the AMCFD.

Task 4 - Start a preliminary product breakdown structure with MEE

A product breakdown structure is an effective tool that details the physical components of a particular product, or system, under consideration. The formal PBS comes in the form of a hierarchy the begins with the final product at the top of the hierarchy followed by the sub-categorized elements of the product. The product breakdown structure is similar to the work breakdown structure. Like the work breakdown structure, a product breakdown structure serves to reduce a complex project into manageable components. As a result, teams can obtain a clear understanding of a product, its components, and what is required to provide those components. This will make the project much more portable when the system is taken by a larger company.

Task 5 - At least one sensor & actuator functioning according to some schedule

The schedule will operate on the seeding phase first as it is the most probable autonomous phase we will keep in the system, since harvesting may not be built into the functionality of the system. We may demonstrate harvesting as a POC for future teams. The seed farm from the simulator will be significantly smaller in demonstrating the actuator since the presentation time will be confined and the actuator moving to plant the seeds would take too long on an actual real-life representation of the farm.

Task 6 - Full simulated farm with well-defined input parameters

We will get well defined & expected real-life settings to run a simulation of our system through all phases including demonstrating error handling. The simulated farm will have scaled values to demonstrate in class and we will run the simulation on the board with real expected values for plant growth and monitoring. This could take ten or more days to evaluate.

Task 7 - Configuration file for all phases in system to run in simulation

Before the simulation is run we will have defined a configuration syntax for every phase in the system and have that syntax validated in the parser. This configuration file will be tested via the simulation with real-life values and scenarios.

Some real-life scenarios could be: astronaut pauses the system, astronaut changed values in config, system actuators down, system sensors down, etc.

Task 8 - Testing Document inheriting from test plan from milestone 1

The testing plan from milestone 1 will be expounded upon with the following:

- **Static** - Done by reading code through formal reviews
- **Functional** - Verifies that each function of the software application operates in conformance with the requirement specification
- **Structural -** Whitebox testing, testing trigger conditions, etc
- **Performance -** Responsiveness and stability under a particular workload
- **Testing scripts -** Automated testing with real-life scenarios

We will also use SonarQube to review, with static analysis, our code to detect bugs, code smells and security vulnerabilities.

Date(s) of meeting(s) with Client during the current milestone 2:

Every two weeks starting 9/13/19

Project Logs			
Action	Date	Time spent in hours	Comments
Met with Dr Chan about config design & readability	10/18/19	1	config design & readability
Met with Dr Chan about Milestone 2 evaluation	10/25/19	1	config design
Met with Dr. Chan for final Milestone 2 evaluation	10/28/19	1.5	

"See Faculty Sponsor Feedback below"

Faculty Sponsor feedback on each task for the current Milestone:

- ### Task 1 - Create state diagram for the system:

- Task 2 - Implement inheritable high-level interfaces/classes for sensor/actuator communication:

- Task 3 - Create simulated farm for testing:

- Task 4 - Create a configuration file for at least one phase in system and run in simulation:

Faculty Sponsor Signature: ________________________________ Date: ________

Faculty Sponsor Evaluation:

1. Faculty Sponsor: detach and return this page to Dr. Chan (HC 322)
2. Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Christopher Millsap	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Giampiero Corsbie	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Faculty Sponsor Signature: ________________________________ Date: __________