COOPERATIVE ROBOTICS FOR SEMI-AUTONOMOUS FARMING

George A. Ditzel III*
Graduate Student, M.S./Computer Science Program, Rivier University

Abstract

The objective of this capstone project was to develop a Proof-of-Concept of two cooperative robots that coordinate with each other to perform some semi-autonomous farming operations. The two robots were to coordinate picking fruit from a grove of trees. One was to be a basket handler (holding the fruit) and the other the fruit picker (picking the fruit). Not all functions defined during analysis were implemented in the timeframe of this project. They were defined to allow the author to further experiment on this topic using the results of this project as a platform for further experiments.

Introduction

The objective of this capstone project was to develop a Proof-of-Concept of two cooperative robots that coordinate with each other to perform some semi-autonomous farming operations. The two robots were coordinate picking fruit from a grove of trees. One will be a basket handler (holding the fruit) and the other will be the fruit picker (picking the fruit).

The disciplines demonstrated in the project are: system requirements development; systems analysis and design; systems construction and programming; system testing, and system prototype demonstration. The project is performed using an Agile/Spiral methodology where the development activity is split into multiple two-week sprints. The first sprint consists of the system requirements development. The other sprints consist of iterations through: system analysis and design; system prototype construction (building and/or programming), and system prototype testing.

The robots have been based on the LEGO Mindstorms EV3 robotics kits that allow for versatile changing of the robots during the Proof-of-Concept build-a-little, test-a-little development.

Definitions

The following definitions are used to provide the context of this paper within the product development phases when using the spiral development model:

- A **Proof of Concept (PoC)** is used to demonstrate certain concepts or theories, and their potential for real-world applications. A PoC is designed to determine feasibility but does not represent deliverables. It is a working concept that may be constructed using a breadboards or evaluation boards, and it is not a complete system or product.
- A **Prototype** is used to evaluate new reference designs that represent a working system model rather than a theoretical one. A Prototype is designed to determine hardware and software feasibility but may not represent deliverables. The hardware prototyping represents a reference design not intended to be a minimal viable product. Depending on how developed, the firmware prototyping may represent deliverable firmware. A Prototype can be considered as a working concept that may be constructed using reference design boards. It is not a complete system or product.

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

- A **Minimum Viable Product (MVP)** is new product developed with enough features to satisfy early adopters. These features should be derived from the PoC and/or Prototyping efforts. The final, complete set of features is only designed and developed after considering feedback from the product's initial users.

This paper focuses on the Proof-of-Concept phase.

Project Plan

The project planning used in this project is a combination a Spiral and Agile process model (see Fig. 1).

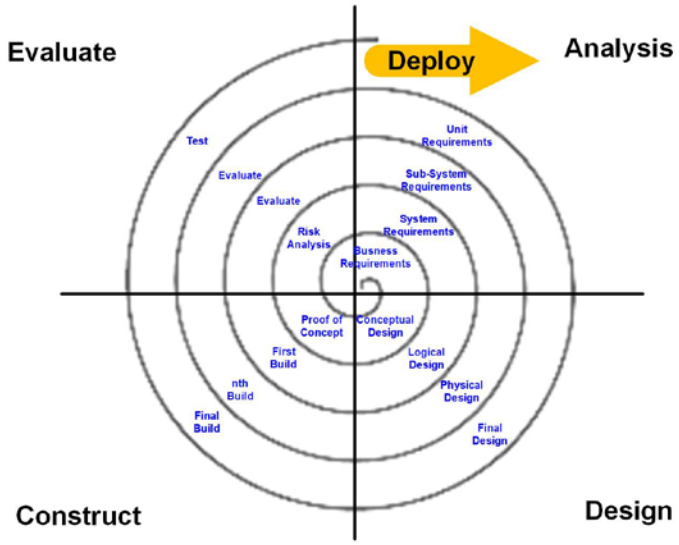

Figure 1: Spiral Process Model

The Spiral process planning is divided into several agile sprints. These sprints are organized into a set of 6 two-week sprints (illustrated in Fig. 2).

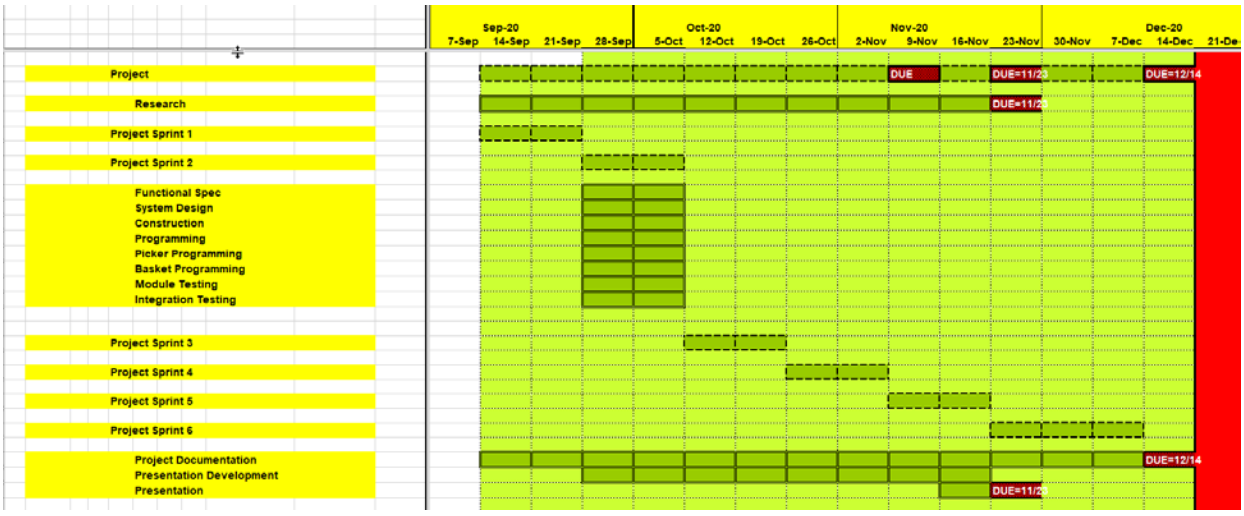

Figure 2: Project Ghant Chart

Each sprint was planned to provide a portion of the Build-a-Little, Test-a-Little spiral process in each sprint.

Functional Specification

Analysis of the system was accomplished through iteratively identifying the essential requirements for the system under development. This method developed enough of the system requirements to allow the design and construction of a portion of the system in an iteration.

The business requirements for autonomous faming vehicles are the following:

- Replace steering wheels and similar manual controls with touch screens that are immediately intuitive;
- Autonomous farm equipment needs to operate continuously for a 24-hour period without direct operation;
- Autonomous equipment needs to monitor and analyze crops detecting plant diseases before symptoms become visible;
- Connectivity among autonomous machines and computers, is necessary;
- Tractors will drive with no farmer in the cab, and specialized equipment will be able to spray, plant, plow, and weed cropland.

Not all functions defined during analysis were implemented in the timeframe of this project. They were defined to allow the author to further experiment on this topic using the results of this project as a platform for further experiments.

System Context

The Proof-of-Concept (PoC) of a Semi-Autonomous Cooperative Robotic Farming system is illustrated in the system context diagram shown in Fig. 3.

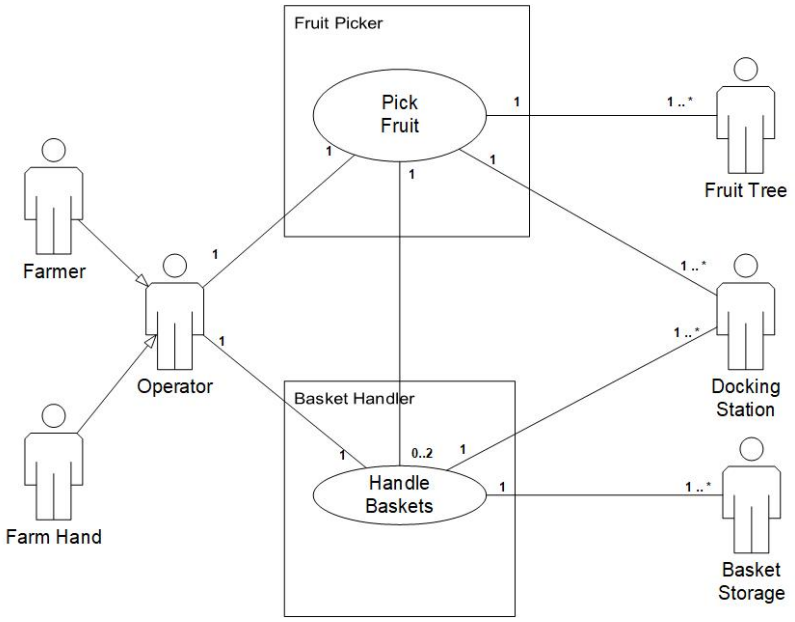

Figure 3: Semi-Autonomous Cooperative Robot Context

This Semi-Autonomous Cooperative Robotic Farming system PoC consists of two subsystems: the Fruit Picker Robot, and one or more Basket Handler Robots.

The actors of the Semi-Autonomous Cooperative Robotic Farming system are:

- **Operator:** The operator of the Robots. For safety reasons only one Operator at a time will be allowed to interact with a robot.
- **Farmer:** The Farmer is an operator and owner of the robots. The Farmer should be an operator with administrative privileges;
- **Farm Hand:** The Farm Hand is also an operator of the robots. The Harm Hand may only have user privileges;
- **Fruit Tree:** Fruit trees in an Orchard that the Fruit Picker robot interacts with to obtain fruit;
- **Basket Storage:** Location where the Basket Handler robots Pick up empty baskets and drops off full baskets;
- **Docking Station:** The station where the robots dock for maintenance and charging;

Use Case Diagrams

Use case diagrams illustrate the primary and secondary functions of a system. Use cases provide an external black-box view of the system and its environment. The use case diagram for the Fruit Picker subsystem is shown in Fig. 4.

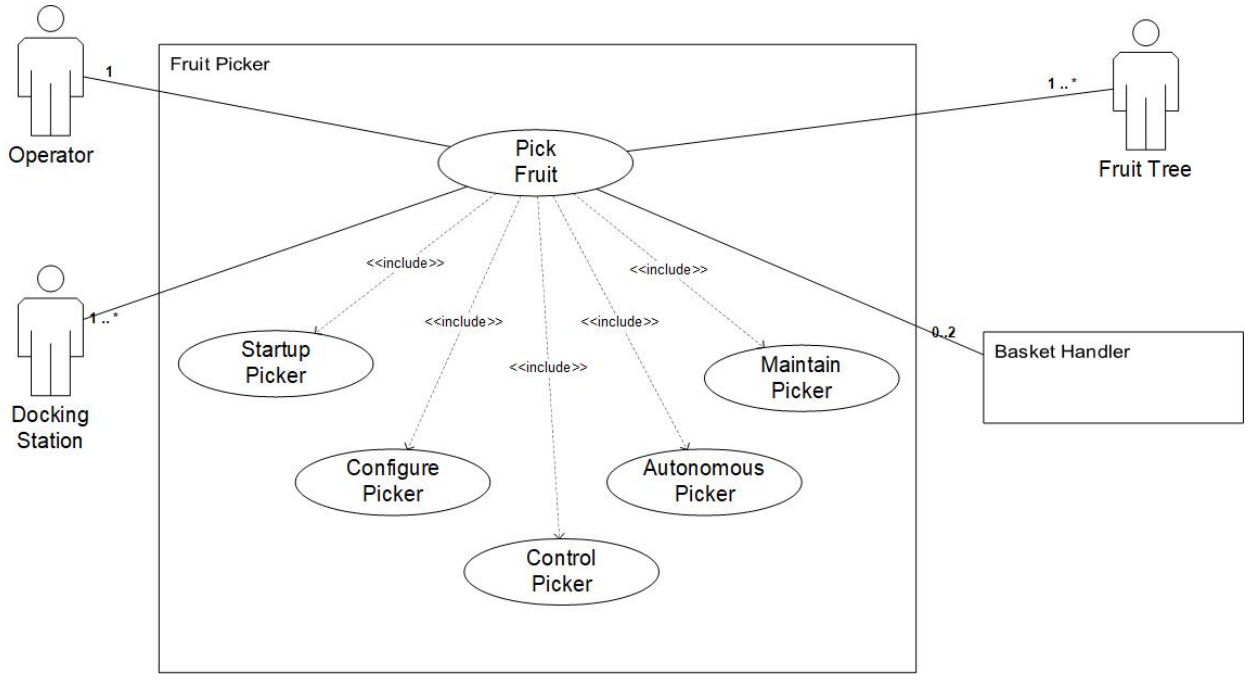

Figure 4: Pick Fruit Use Case

There is an *include* relationship between Pick Fruit use case and the other use cases: Startup Picker, Configure Picker, Control Picker, Autonomous Picker, and Maintain Picker. The autonomous operations of the Fruit Picker are further illustrated in Fig. 5.

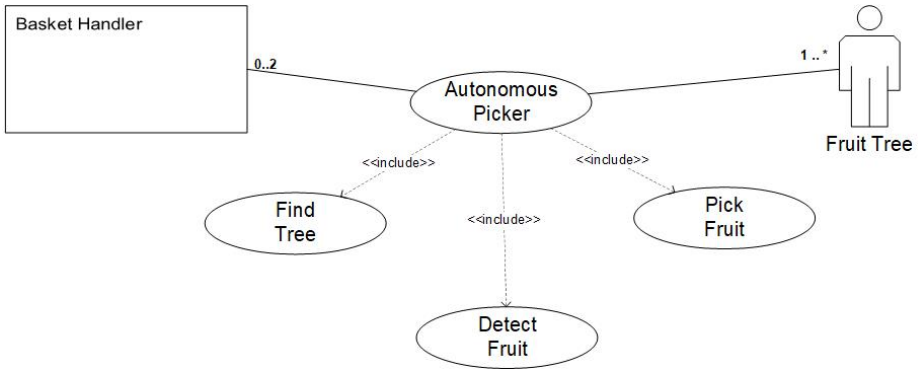

Figure 5: Autonomous Picker Use Case

The Handle Basket use case describes the main functions for the Fruit Picker subsystem (see Fig. 6).

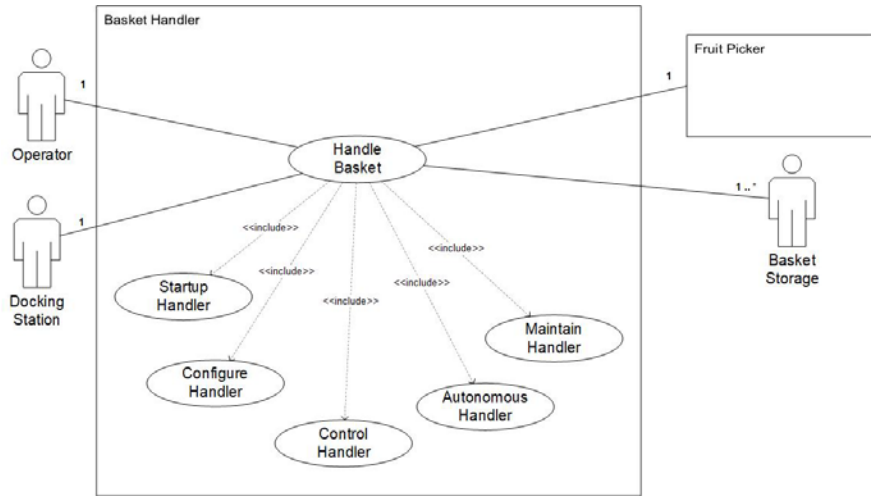

Figure 6: Handle Basket Use Case

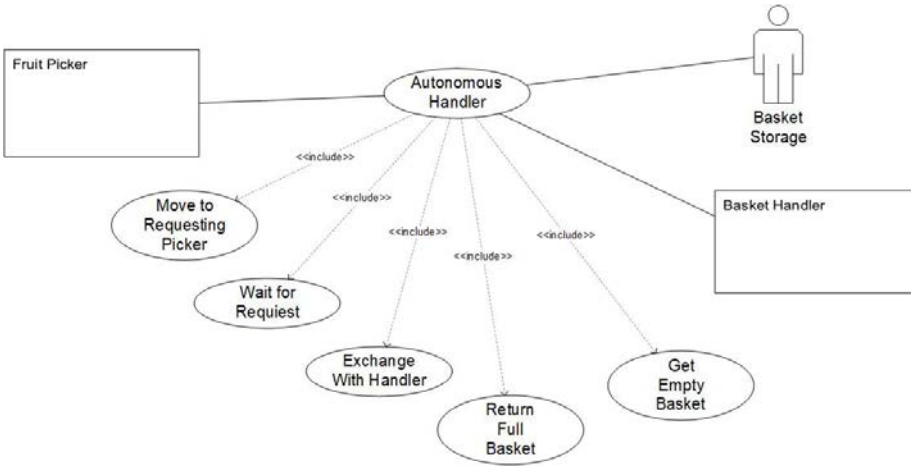

Figure 7: Autonomous Handler Use Case

There is an *include* relationship between Handle Basket use case and the other use cases: Startup Handler, Configure Handler, Control Handler, Autonomous Handler, and Maintain Handler. The autonomous operations of the Fruit Picker are further illustrated in Fig. 7.

State Machine Diagram

A Flow of Events describes one or more paths through a use case which is composed of sequence of operations in which several objects collaborate to achieve the purpose of the use case. The flow of events for the Pick Fruit use case are illustrated in the state machine shown in Fig. 8.

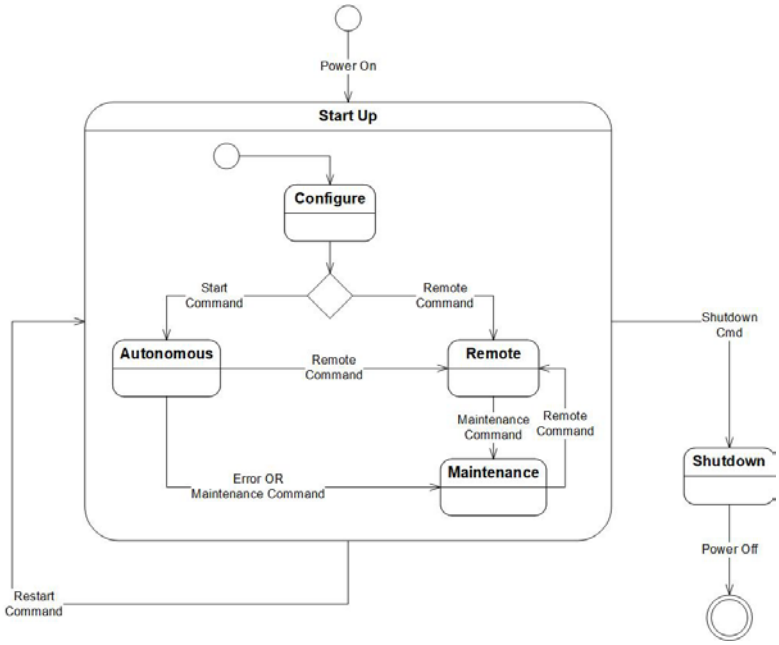

Figure 8: Picker Robot State Machine

Design

The system design was iteratively performed, identifying the solution consistent with the system requirement. The partial ("enough") design of the system during an iteration allowed the construction and programming of a portion of the system.

Hardware Design

The hardware design involved the design of a common motion platform used in two robot types, the Fruit Picker robot and the Basket Handler robot.

The Motion platform consists of the following hardware components (see Fig. 9): 1) The Controller, which is design to be a composite of the controller processor, the Bluetooth communication interface, and the User Interface, itself a composite of the display and the push-buttons, and 2) The aggregated Motor, a set of two entities.

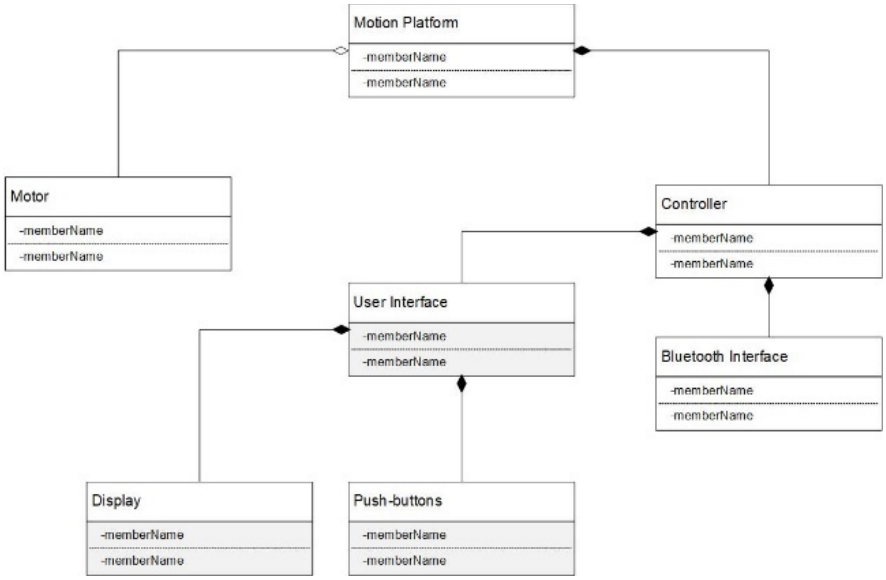

Figure 9: Common Motion Platform Design

The Fruit Picker robot is described as a Motion Platform aggregated with the following components (see Fig. 10): 1) The IR Beacon that is used to allow the Basket Handlers to determine where the Fruit Picker is, and 2) The Fruit Picking Arm for picking the fruit. The arm itself is an aggregate of the Light Sensor and Ultrasonic Sensor.

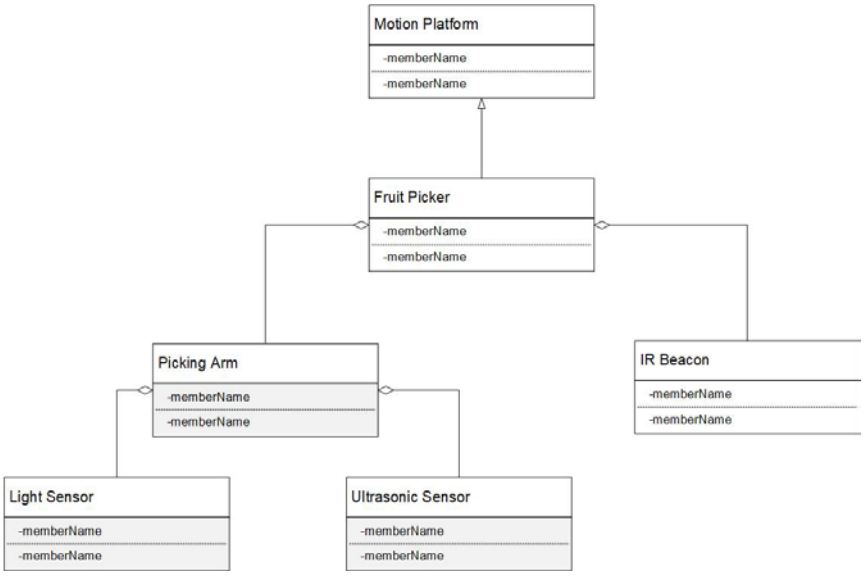

Figure 10: The Fruit Picker Robot Design

The Basket Handler robot is described as a Motion Platform aggregated with the following components (see Fig. 11): 1) An IR Sensor that is used to determine where the Fruit Picker is, and 2) A Touch Sensor, to determine if a Basket has been mounted on the handler.

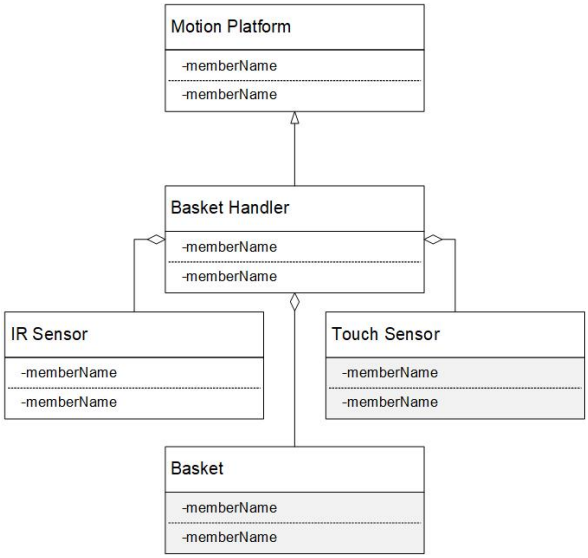

Figure 11: The Basket Handler Robot Design

Firmware Design

The firmware design was modelled referencing the Erlang Robotic Framework [1] (see Fig. 12). The portions of the model have been used is each robot.

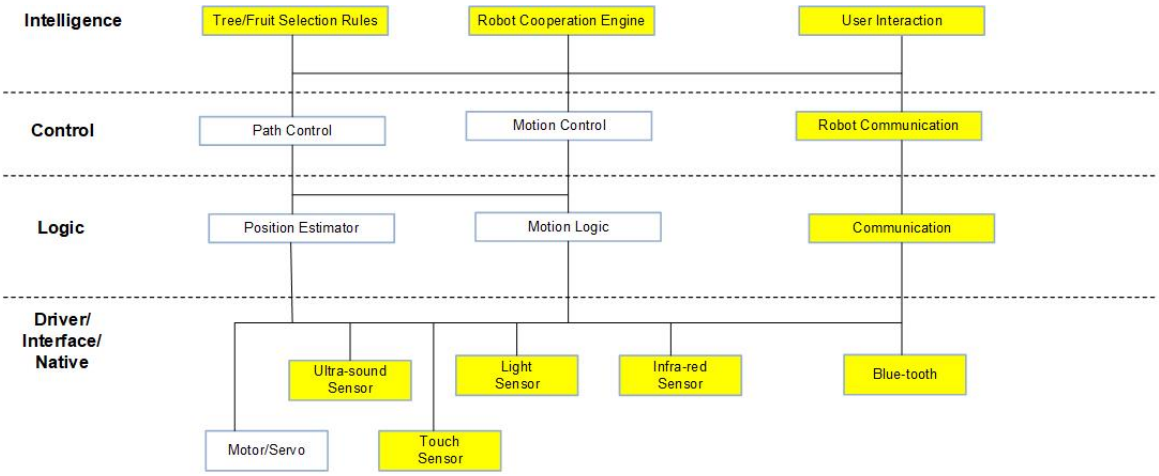

Figure 12: Subsystem Firmware Design

The Fruit Picker and Basket Handler robot firmware consists of the following layers and components:

- The Driver/Interface/Native layer provides a driver for each specific peripheral. This layer consists of the following components:
  - Motor/Servo Drivers, used by the Logic layer to move the robot
  - Touch, Light, Infra-Red, Ultra-sound, Infra-red sensor drivers, used to sense the environment
  - The Blue-tooth communication drivers.

- Logic Layer translates device-specific functions to application-specific functions. This layer consists of the following components:
  - Position Estimator, Motion Logic
  - Communication Messaging.
- The Control layer implements feedback loops required to ensure that actions are performed as desired. This layer consists of the following components:
  - Path Control
  - Motion Control
  - Robot Communication.
- Intelligence Layer implements and controls robot strategy using high-level rules. This layer consists of the following
  - Tree/Fruit Selection Rules
  - Robot Cooperation Engine
  - User Interaction.

Construct

This section describes the producing of the various components needed to construct the system, both hardware and software:

- **Construction** – Construct enough of the system hardware, using the LEGO Mindstorms EV3 robot construction kits, to allow for programming of a portion of the system.
- **Programming** – Program enough of the common portions of the system to allow for robot programming and testing of a portion of the system.
- **Picker Programming** – Programming enough of the Fruit Picker Robot to allow the module and integration testing of a portion of the system.
- **Handler Programming** – Programming enough of the Basket Handler Robots to allow the module and integration testing of a portion of the system.

The Robots

The robots were constructed using elements from the LEGO Mindstorms EV3 robot development kit. The motion platform (see Fig. 13) was assembled with LEGO Technics bricks, two large motors, and the EV3 computer brick.

Figure 13: Motion Platform Picture

The Fruit Picker Robot (see Fig. 14) was constructed with LEGO Technics bricks, two large motors, the EV3 computer brick, and the IR Beacon. The Picker Arm was not constructed for this PoC.

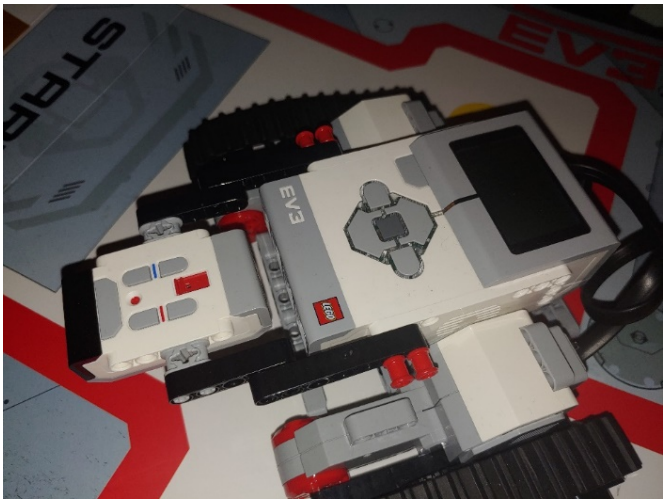

Figure 14: The Fruit Picker Robot

The Basket Handler Robot (see Fig. 15) was constructed with LEGO Technics bricks, two large motors, and the EV3 computer brick and the IR Sensor. The Basket element was not constructed for this PoC.

Figure 15: The Basket Handler robot

Programming Environment and Tools

The programming environment and tools (SDK) used to develop the robot programs is the LEGO Mindstorms EV3 Home Edition. This SDK can be found at the LEGO website:

https://www.lego.com/en-us/themes/mindstorms/downloads. You can find the EV3 Software PC application (see Fig. 16) there.

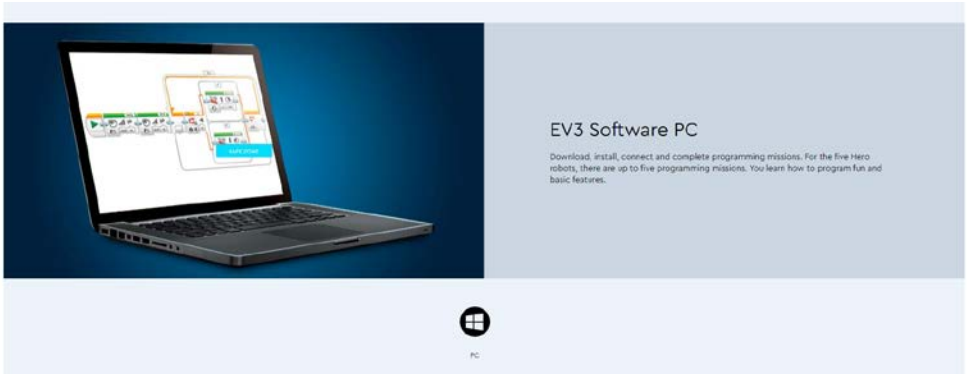

Figure 16: The EV3 Software PC Application

The development environment for the Home edition is shown in Fig. 17.

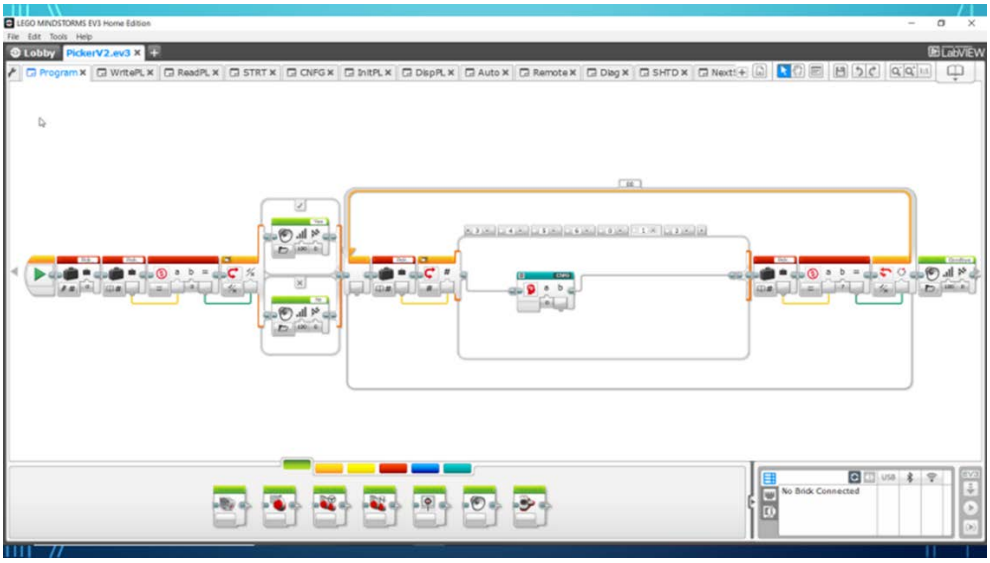

Figure 17: LEGO Mindstorms EV3 program development environment

Evaluation

The system components were iteratively evaluated, tested, and integrated as they were constructed. A plan was initially developed with two iterative levels: Module Testing and Integration Testing.

The Fruit Picker Module Testing consisted of the following hardware construction and firmware implementations:

- Motion Platform Construction with the following firmware implementations:
  - Fruit Picker main program
  - State Machine program

- Motion Control program
- Path Control program.

- Picker Arm Construction with the following firmware implementations:
  - Light Sensor program
  - Ultrasonic Sensor program
  - Arm Control program
  - User Interface program.

The Basket Handler Module Testing consisted of the following hardware construction and firmware implementations:

- Motion Platform Construction with the following firmware implementation:
  - Basket Handler main program;
  - State Machine program
  - Motion Control program
  - Path Control program.
- Basket Construction with the following firmware implementations:
  - Touch Sensor program.

The System Integration Testing consisted of the following hardware construction and coordinated robot interaction:

- Fruit Picker/Basket Handler Bluetooth Communication
  - Request a basket
  - Basket nearing full
  - Replace basket
- Picker/Basket IR Communication
  - Basket Hander finds Fruit Picker
- Synchronized Picker/Basket Path Control
  - Coordinated Fruit Picking.

Iteratively evaluating, testing, and integrating the system components as they are constructed. The module/ Integration testing completed in the PoC are marked with **Done**.

Table 1: Module/Integration Test Results

Module/Integration Test	**Evaluate**
Fruit Picker Module Test	
Motion Platform	
Main Program	**Done**
State Machine	**Done**
Motion Control	**Done**
Path Control	**Done**
Picker Arm	
Basket Handler Module Test	
Motion Platform	
Main Program	**Done**
State Machine	**Done**

		Motion Control	**Done**
		Path Control	**Done**
		Basket	
	System Integration Test		
		Picker/Handler Bluetooth Comm	
		Request Basket	**Done**
		Picker/Handler IR Comm	**Done**
		Handler finds Picker	**Done**
		Sync Picker/Handler Path Control	**Done**
		Coordinated Fruit Picking	**Done**
		Return to docking stations	**Done**

Demonstration

The demonstration of the completed portions of this Proof-of-Concept was performed in the classroom on November 23, 2020.

Issues for Future Studies

More semi-autonomous functionality was defined then could be implemented in the timeframe of this 14-week class project. An attempt was made to complete the motion platform functionality and some robot interaction features for the demonstration.

Based on this experience, the following issues were discovered: 1) The programming environment used lacked object-oriented programming capabilities, requiring rethinking procedurally during programming, and lacked recognizable debugging features, which required alternative debugging approaches, and 2) The sensors that came along with the development kits had less sensor data resolution then was expected.

Future studies could continue the Proof-of-Concept efforts as follows: 1) More detailed, object-oriented programming could be done using Python (it was found out late in the project that it may be possible to include a Python SDK in the EV3 development environment), 2) Complete the Picker Arm using both the Light and Ultrasonic sensors, performing sensor fusion and machine learning operations to detect and pick appropriate fruit, and 3) Complete the basket and the basket handling.

Conclusion

Developing semi-autonomous robot requires more involvement then can be done working part-time for 14 weeks. The Proof-of-Concept was the interesting approach in developing as much of the robots as could be done during this short time period. There requires much more involvement then could be applied at this time. More work is needed in developing algorithms to detect and determine “pickable” fruit, possibly in the area of machine learning. The robots would also benefit from algorithms for path projection and object avoidance.

References

[1] Corrado Santoro. An Erlang Framework for Autonomous Mobile Robots. In *Proceedings of the 2007 ACM SIGPLAN Workshop on Erlang*, Freiburg, Germany, October 5, 2007. DOI: 10.1145/1292520.1292533.

Sites

The programming environment and tools (SDK) used to develop the robot programs is the LEGO Mindstorms EV3 Home Edition. This SDK can be found at the following website: https://www.lego.com/en-us/themes/mindstorms/downloads.

* **GEORGE A. DITZEL III** is a graduate student at Rivier University. He will be completing his Master's degree in Computer Science in spring 2021. Prior to that he had obtained a B.S. in Computer Engineering from Spring Garden College, Chestnut Hill, PA. This work (based on his final capstone project in the CS699A Professional Seminar in fall 2020) has inspired his work in robots at his company along with an interest in developing solution to support social distancing during the COVID-19 pandemic.