Software Formalization

Year: \_\_22\_\_ Semester: \_FALL\_ Team: \_12\_\_ Project: \_\_\_\_\_R.A.C.H.E.L.\_\_\_\_\_\_\_\_\_\_

Creation Date: ­\_\_\_\_\_\_\_September 28, 2022\_\_\_\_ Last Modified: October 1, 2022

Author: \_\_\_\_\_\_\_\_James Hubbard\_\_\_\_\_\_\_\_\_\_ Email: \_\_hubbar31@purdue.edu\_\_\_\_

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Third Party Software** |  | x2 |  |  |
| **Description of Components** |  | X3 |  |  |
| **Testing Plan** |  | x3 |  |  |
| **Software Component Diagram** |  | x4 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

*Relevant overall comments about the paper will be included here*

1.0 Utilization of Third Party Software

|  |  |  |
| --- | --- | --- |
| Name | OpenNI2 | OpenCV |
| Description | OpenNI2 is a library that facilitates communication with PrimeSense compatible camera sensors. | OpenCV implements hundreds of computer vision techniques, primarily focusing on techniques that could be described as “signal processing” instead of “artificial intelligence”. |
| Intended Use | OpenNI2 is being used in our project in order to interface with the ASUS Xtion Pro LIVE camera above our table. The functionality of OpenNI2 allows the team to access the raw data coming out of the camera sensor and pass it directly to OpenCV. | OpenCV will be used in our project to track a ping pong ball, and also to detect gestures, both of which are stretch goals. |
| License | Apache 2.0 [1] | Apache 2.0 [2] |
| Notes | Code can be modified, redistributed, and reused freely. Any original copyright/attribution notices must remain witth the code unless they are irrelevant to the derivative work. [3] | See left. |

|  |  |  |
| --- | --- | --- |
| Name | SCons | SW4STM32 |
| Description | SCons is a build system not unlike CMake, except that it is controlled by a Python program instead of a CMakelists or similar file. | System Workbench for STM32 is an all-in-one IDE for working with STM32 family microcontrollers. |
| Intended Use | SCons is being used for the build of every part of our software which lives on the laptop. | SW4STM32 is being used to build our microcontroller code, and program it onto hardware. This tool also allows for step-by-step debugging of the running microcontroller. |
| License | MIT License [4] | GPLv3 and ST Liberty Licenses |
| Notes | Code can be modified, redistributed, and reused freely. The original copyright notice must be redistributed with the software or any significant portions. [3] | ST-provided firmware is licensed under the ST Liberty license, which places no restrictions on the code that the end user writes. [5] |

2.0 Description of Software Components

Please refer to the diagram in Appendix 1 to supplement the following tables.

Source code can be found in the repositories at: <https://github.com/purdue-RACHEL>

| Component | Functions | Provided Interfaces | Required Interfaces |
| --- | --- | --- | --- |
| CameraInterface  (developed by the team) | readDepth  readDepthSlice  readColor | Camera frames (depth and color) from the PrimeSense sensor.  Depth frames come in two formats, the full 16-bit image, or an 8-bit slice from somewhere within the 16-bit image. | Device, VideoFrameRef, VideoMode, and VideoStream from OpenNI2, and  Mat from OpenCV. Used as-is. |
| BallDetector  (developed by the team) | locateBall2D  locateBall3D | Ball positions inferred from camera data.  Ball positions may be expressed as their 2D “shadow” the table, or their 3D position above the table. | Camera frames from the PrimeSense sensor, provided by CameraInterface  Mat from OpenCV, used as-is |
| TTTable  (developed by the team) | positionTableManual  positionTableInfer  cam2table  table2cam | Position and orientation of the table.  Transforms camera-relative coordinates to table-relative coordinates and vice versa.  Shall determine position/orientation from user input or by inferrring from camera data | Camera frames from the PrimeSense sensor, provided by CameraInterface  Mat from OpenCV, used as-is |
| GameObject  (developed by the team)  Parent class to each game we develop | determinePoint  displayLoop | Determines points scored using any, all, or none of its required interfaces.  Displays graphics and the state of the game to the projector | Ball position from BallDetector  Coordinate transformations from TTTable  Bounce detections from microcontroller via UARTReceiver |
| UARTReceiver  (developed by the team) | read  write | Communication with the microcontroller from the laptop | Serial connection with the microcontroller to interface with uart.c |
| uart.c  (developed by the team) | setup\_uart  send\_packet | Communication with the laptop from the micrcontroller  Setup function to configure microcontroller control registers. | Hardware timers and UART |
| adc.c  (developed by the team) | setup\_adc  start\_adc\_channel  setup\_tim6  read\_adc  set\_threshold | Bounce detections from the microphones attached to our table  Bounce detections can be expressed either as which side of the table was hit, or a position determined by triangulation  Setup function to configure microcontroller control registers. | Hardware timers and ADC |
| keypad.c  (developed by the team) | setup\_keypad  set\_row  setup\_tim7 | Keypresses from the keypad  Setup function to configure microcontroller control registers. | Hardware timers and GPIO pins |
| main.c  (developed by the team) | main | None, microcontroller functionality is interrupt-driven | Setup functions from each of the microcontroller software components  Communications with the laptop from the microcontroller |

3.0 Testing Plan

| Component (Priority) | Required Tests |
| --- | --- |
| CameraInterface  (7) | The core fucntionality of CameraInterface is providing camera data in the form of frames that integrate easily with OpenCV. It must be tested to ensure that the returned images reflect what’s actually happening in front of the camera. It must also be tested for its ability to acquire the camera successfully and handle errors like a lack of connected cameras. |
| BallDetector  (9) | BallDetector must be tested on how well it can distinguish a ball from a hand, debris on the table, reflections, and projected images. A range of conditions will need to be tested to ensure that the ball is correctly identified among all these sources of noise. Bright colors will be emitted from the projector, objects other than balls will be left on the table, etc. during testing procedures |
| TTTable  (8) | TTTable pprovides a visual interface for manually inputting the positon of the table. This interface needs to be tested for usability.  The team must also test that the calculated position and orientation of the table is correct within some margin of error. We will test this by tilting the camera and table and repeatedly comparing the calculated position and orientation with the real position of the table. |
| GameObject  (4) | GameObject must be tested with many different combinations of its inputs to ensure that it awards points correctly in all the situations which would reasonably arise in a game of table tennis. Each game for our system, including standard table tennis, will have its own GameObject which will be individually tested for correctness, using both software testing tools and real games played by the team. We will use code coverage metrics to make sure that we have tested every rule that we’ve specified in each game |
| UARTReceiver  (3) | UARTReceiver must be tested to ensure that the data being written to the serial port is as intended, and that the data being read corresponds with the data on the bus. This means using an oscilloscope to read the bytes being sent back and forth and verify their accuracy. |
| uart.c  (2) | See UARTReceiver above. Uart.c must also be tested to make sure that it correctly responds to a reset message from the laptop by resetting the micrcontroller |
| adc.c  (1) | This component contains the bulk of the functionality performed by the micrcontroller. Tests will evaluate the component’s ability to:  Correctly identify the side of the table on which a ball bounced  Correctly identify the region in which the ball bounced using triangulation  Discern ball bounces from other vibrations caused by environmental noise or jostling of the table  Correctly relay bounce information to uart.c  These tests will al be performed in the SW4STM32 debugger |
| keypad.c  (5) | This component must be tested to ensure that every key on the keypad is correctly and timely identified. Tests must also verify that the data being relayed to uart.c is accurate using the SW4SYM32 debugger |
| main.c  (6) | main is really only responsible for signaling to the other components when they should get ready to begin. It will be stepped through in the SW4STM32 debugger to check that it calls each component’s setup function successfully. |

4.0 Sources Cited:

[1] Structureio, “Structureio/openni2: Openni2,” GitHub. [Online]. Available: <https://github.com/structureio/OpenNI2>. [Accessed: 01-Oct-2022].

[2] Opencv, “Opencv/opencv: Open source computer vision library,” GitHub. [Online]. Available: <https://github.com/opencv/opencv>. [Accessed: 01-Oct-2022].

[3] “Licenses,” Choose a License. [Online]. Available: <https://choosealicense.com/licenses/>. [Accessed: 01-Oct-2022].

[4] S. C. Foundation, “Frequently asked questions,” SCons, 15-Sep-2021. [Online]. Available: <https://scons.org/faq.html>. [Accessed: 01-Oct-2022].

[5] “Ultimate Liberty Software License Agreement” STMicroelectronics, 23-Feb-2016. [Online]. Available: <https://www.st.com/content/ccc/resource/legal/legal_agreement/license_agreement/group0/0d/63/cc/f0/70/b6/47/a7/DM00601279/files/DM00601279.pdf/jcr:content/translations/en.DM00601279.pdf>. [Accessed: 02-Oct-2022].

Appendix 1: Software Component Diagram

