Software Formalization

Year: 2023 Semester: Fall Team: 16 Project: Air Hockey Robot

Creation Date: September 28, 2023 Last Modified: September 30, 2023

Author: Abby Haluska Email: ahaluska@purdue.edu

1.0 Utilization of Third Party Software

The robot air hockey project warrants software to be written on both a PC and a microcontroller. The software on the PC will handle all image processing and motion prediction. In order to do so, the *OpenCV* and *NumPy* libraries will be leveraged. Additionally, Python’s built-in *time* and *math* modules will be used. The PC will send data to the microcontroller over UART, in which the *PySerial* module and *infi.devicemanager* API will be used to communicate with the microcontroller. The microcontroller will be responsible for translating data received from the PC into instructions for the motor drivers. The microcontroller will also accept input from the IR sensors and drive the 7-segment displays. To accomplish this, STM32Cube’s *Hardware Abstraction Layer (HAL)* will be leveraged. All software dependencies fall under permissive style software licenses that permit free commercial use of the software. Table 1 details the usage cases of each software dependency.

Table 1: Third Party Software

| **Dependency** | **Licensing** | **Description** | **Usage Case** |
| --- | --- | --- | --- |
| *Python Libraries* | | | |
| OpenCV | Apache 2 License | OpenCV is an open source computer vision library that offers easily accessible image processing modules, video analysis modules, object detection modules, and other functionality to be used in image filtering and object tracking algorithms. This library is free under the Apache 2 License. It is to be used with Python. [1] | OpenCV will be used to fulfill the team’s software goal of estimating the position and trajectory of a puck in motion using computer vision. |
| NumPy | BSD License | NumPy is a Python library to be used in scientific computing. This library offers multidimensional array objects and various functions for operating upon arrays. [2] | NumPy will be used in object tracking in which images of a puck in motion will be translated into NumPy arrays for further processing. |
| *Python Modules / APIs* | | | |
| PySerial | Free License under Chris Liechti | PySerial is a Python module that offers access to serial ports by running backends on Windows, OSX, Linux, and BSD systems. [3] | PySerial will be used in sending image processing data to the microcontroller over USB-to-UART. |
| infi.devicemanager | BSD License | infi.devicemanager is a module that offers Python binding to the device manager’s API on any windows system. [4] | This module will be used in detecting which port the USB-to-UART IC is plugged into before sending serial data from PC to microcontroller. |
| time | PSF License Agreement | The time module offers a variety of time-related functions to be used in real-time applications [5]. | This module will be used in generating delays between each image frame that is processed. |
| math | PSF License Agreement | The math module offers access to common math functions as defined in C. [6] | The math module will be used in computing the projected trajectory of a puck in motion. |
| *STM32Cube Libraries* | | | |
| HAL (Hardware Abstraction Layer) | BSD License | The HAL library is an abstraction layer embedded within STM32Cube’s IDE to offer maximized portability and hidden peripheral complexity. [7] | The HAL library will be leveraged when sending instructions to the microcontroller and the 7-segment display. |

2.0 Description of Software Components

The primary software components of the Air Hockey Robot include a structure storing the projected location of a puck in motion, a structure storing coordinated movement instructions for the motors, a function dedicated to parsing received data and translating that data into motor instructions, and a function to update the scoreboard whenever an IR break beam sensor detects a goal. These components are detailed below.

2.1 Structure Containing Puck and Paddle Coordinates

Upon processing each image frame, the PC will generate an array including the estimated

X and Y coordinates of a puck detected by color, the estimated velocity of a puck in

motion, and the predicted future location of a moving puck. The position vectors will

correspond to pixels on a screen, in which it will be necessary that the camera remains in

a position as close to stagnant as possible. The coordinates of a paddle will also be

included in this structure, in which the paddle will also be identified using color

detection. This software component is critical as it will be used in generating the structure

containing motor instructions. This structure will remain on the PC and will not be sent to

the microcontroller.

2.2 Structure Containing Motor Instructions

Following object detection, the projected coordinates and velocity of a puck in motion

will be combined with the current position of a paddle to identify the adjustments needed to block and attack the moving puck. These adjustments will be written in the form of 8-bit packets, in which the most significant bit corresponds to the direction of movement. The remaining bits are allocated to specify the distance that the motors must move the puck. A unique grid system will be created to accomplish this. This structure will be generated by the PC and will be sent over UART to the microcontroller.

2.3 Parsing Received Data and Sending Instructions

The motor instruction packet will be sent over USART2, in which a global interrupt will

be triggered whenever the microcontroller receives a packet. The team will write code within HAL’s *HAL\_UART\_RxCpltCallback* function to then parse through the packet and update variables correlating to motor speed and direction. This function identifier is referenced from the HAL library, but the contents of the function will be written by the team. If the motor speed and direction variables change within this function, the updated variables will be written as arguments in a custom function *step\_motor(step, dir)* that sets the GPIO pins connected to the *step* and *dir* pins of the motor driver. This function will be written by the team but it will leverage the HAL\_GPIO APIs that are included within STM32Cube’s abstraction layer.

2.4 Updating the Scoreboard whenever a Score is Detected

The IR break beam sensors used in detecting score will be tied to an external interrupt so

that a break in the beam leads to an interrupt service routine identified by the *HAL\_GPIO\_EXTI\_Callback* function. This function identifier is referenced from the HAL library, but the contents of the function will be written by the team. Whenever an interrupt is triggered, a “score” variable will be updated to reflect the new score. This update will then be sent over SPI through a custom *update\_score(score)* function, which uses the *HAL\_SPI\_Transmit* function to send data to the 7-segment display. This function also uses the HAL\_GPIO\_WritePin function to pull the latch pin of an HC595 HIGH so that the outputs update appropriately.

Software component diagrams outlining the details above can be found below in Appendix 1.

3.0 Testing Plan

To ensure a smooth integration process, all individual components will be tested independently before robot construction. Once a component is prototyped, the following protocol will be followed to simplify debugging and validate functionality.

Given that the first two software components are structures versus algorithms, minimal testing will need to take place to ensure that these components are ready for project integration. To test the first two structures, the following steps will be taken:

3.1 Structure Containing Puck and Paddle Coordinates

1. Connect camera to PC
2. Run python script
3. Hit puck across air hockey table
4. Verify: an array including the estimated X and Y coordinates of a puck detected by color, the estimated velocity of a puck in motion, and the predicted future location of a moving puck is printed to the terminal for every frame analyzed.

3.2 Structure Containing Motor Instructions

1. Connect camera to PC
2. Run python script
3. Hit puck across air hockey table
4. Verify: a motor instruction 8-bit packet containing direction and distance data is printed to the terminal for every frame analyzed.

3.3 Parsing Received Data and Sending Instructions

A *UART\_transmit* function call will be commented out in the

*HAL\_UART\_RxCpltCallback* function to assist in debugging. This statement will transmit

the received data to the terminal for verification of the UART protocol.

To confirm that the microcontroller is receiving the motor instruction data from the PC,

the following steps can be taken:

1. Open project in STM32Cube and upload script to microcontroller
2. Uncomment the *UART\_transmit* function call in the *HAL\_UART\_RxCpltCallback* function
3. Open a terminal window in STM32Cube’s IDE connected to the correct serial port with baud rate of 115200
4. Connect camera to PC
5. Ensure motors are powered and connected
6. Run python script
7. Hit puck across air hockey table
8. Refresh terminal
9. Verify: a motor instruction 8-bit packet is printed to the terminal for every frame analyzed
10. Verify: the motors move when the program is ran

3.4 Updating the Scoreboard whenever a Score is Detected

To test the EXTI interrupt associated with the IR break sensors and the SPI protocol

associated with the 7-segment display, the following steps can be taken:

1. Open project in STM32Cube and upload script to microcontroller
2. Run script
3. Verify: the 7-segment display reflects a score of 0
4. Place a puck in between one of the IR receiver and transmitter pairs
5. Verify: the 7-segment display updates to a score of 1

Throughout prototyping, individual tests will be conducted regularly to verify the functionality of all software and hardware components. The tests described above are specific to the software components described throughout this report, but are subject to change given software updates.

4.0 Sources Cited:

[1] “Introduction,” OpenCV, https://docs.opencv.org/4.8.0/d1/dfb/intro.html (accessed Sep.

30, 2023).

[2] “NumPy documentation#,” NumPy documentation - NumPy v1.26 Manual,

https://numpy.org/doc/1.26/ (accessed Sep. 30, 2023).

[3] “Pyserial,” pySerial, https://pyserial.readthedocs.io/en/latest/pyserial.html (accessed Sep.

30, 2023).

[4] “Infi.devicemanager,” PyPI, https://pypi.org/project/infi.devicemanager/ (accessed Sep.

30, 2023).

[5] “Time - Time Access and conversions,” Python documentation,

https://docs.python.org/3/library/time.html (accessed Sep. 30, 2023).

[6] “Math - mathematical functions,” Python documentation,

https://docs.python.org/3/library/math.html (accessed Sep. 30, 2023).

[7] STM32Cube includes: Effort, time and cost ... - stmicroelectronics,

https://www.st.com/resource/en/user\_manual/um1785-description-of-stm32f0-hal-and-lo

wlayer-drivers-stmicroelectronics.pdf (accessed Sep. 30, 2023).

Appendix 1: Software Component Diagram

Figure 1A: Software Component Diagram (microcontroller)

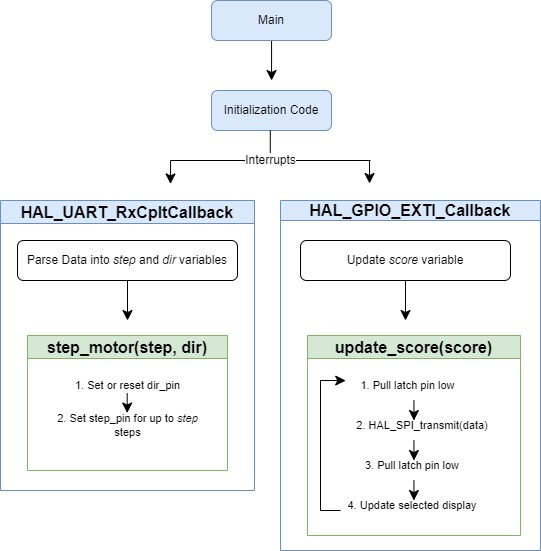
**

Figure 1B: Initialization Diagram (microcontroller)

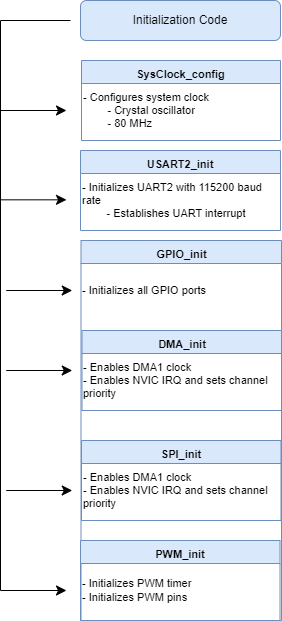


Figure 2: Software Component Diagram (PC)

A diagram of a software process

Description automatically generated