Detailed Software Design
for
Video-Based Control System for Automatic Standing Desks

Version 1.1

Prepared by Marco Borg, Mohit Kapur, Justin Power, Tristan Grondin, and Constantine Valettas

University of Ottawa

April 12, 2023

Contents

1. Introduction

The software component in the detailed design is to provide sufficient information as to how the system will operate through a created program. Multiple hardware elements will be used throughout this project and will require code to allow each element to be controlled and pass data amongst each other to obtain a common goal. For this project, the software can be split up into different sub-sections where sub-routines can be developed independently and merged into a single program later. This allows team members to be properly organized and encourages them to focus on small parts at a time instead of trying to complete and generate the entire code needed for the project at once. Therefore, the information provided below will be available to all team members so that the software development process can be properly documented and organized.

1.1. Document overview

This document describes the design of the software components, packages, and elements of the microcontroller, actuators, and computer vision devices, part of the Video-Based Control System for Automatic Standing Desks software development project.

1.2. References

1.2.1. Project References

#	Document Identifier	Document Title
[R1]	SRS	**System Requirements Specification for Body Tracking Actuator System**
[R2]	Url Access	**URL Access for Computer Vision Subroutine**

1.2.2. Standard and regulatory References

#	Document Identifier	Document Title
[STD1]	IEEE Std 1016-1998	IEEE Recommended Practice for Software Design Descriptions
[STD2]	IEEE Std 830-1998	IEEE Recommended Practice for Software Requirements Specifications

2. Software Architecture Overview

The software architecture will consist of multiple sub-components that will be made up and implemented into a top-level design. Therefore, the main function will be run in the program which will mark the start of the program every time the system is turned on or initialized. The architecture will be made in a specific manner so that all the sub-components can communicate with each other regardless of the computer language used. This is important because the computer vision system will need to be written in python for simplicity while the rest of the software will need to be written in C+ as the code will be implemented into a raspberry pi during the visualization part of the project. Thus, the chart shown in Figure 1 can illustrate how the physical circuit will be developed and mark what sub-systems need to communicate with one another to show the top-level hierarchal software design.

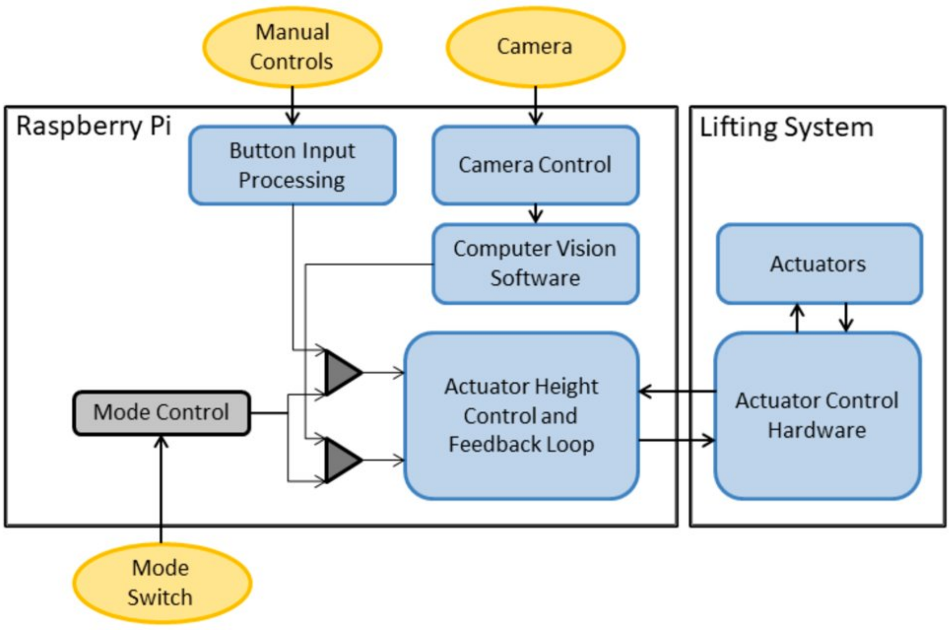
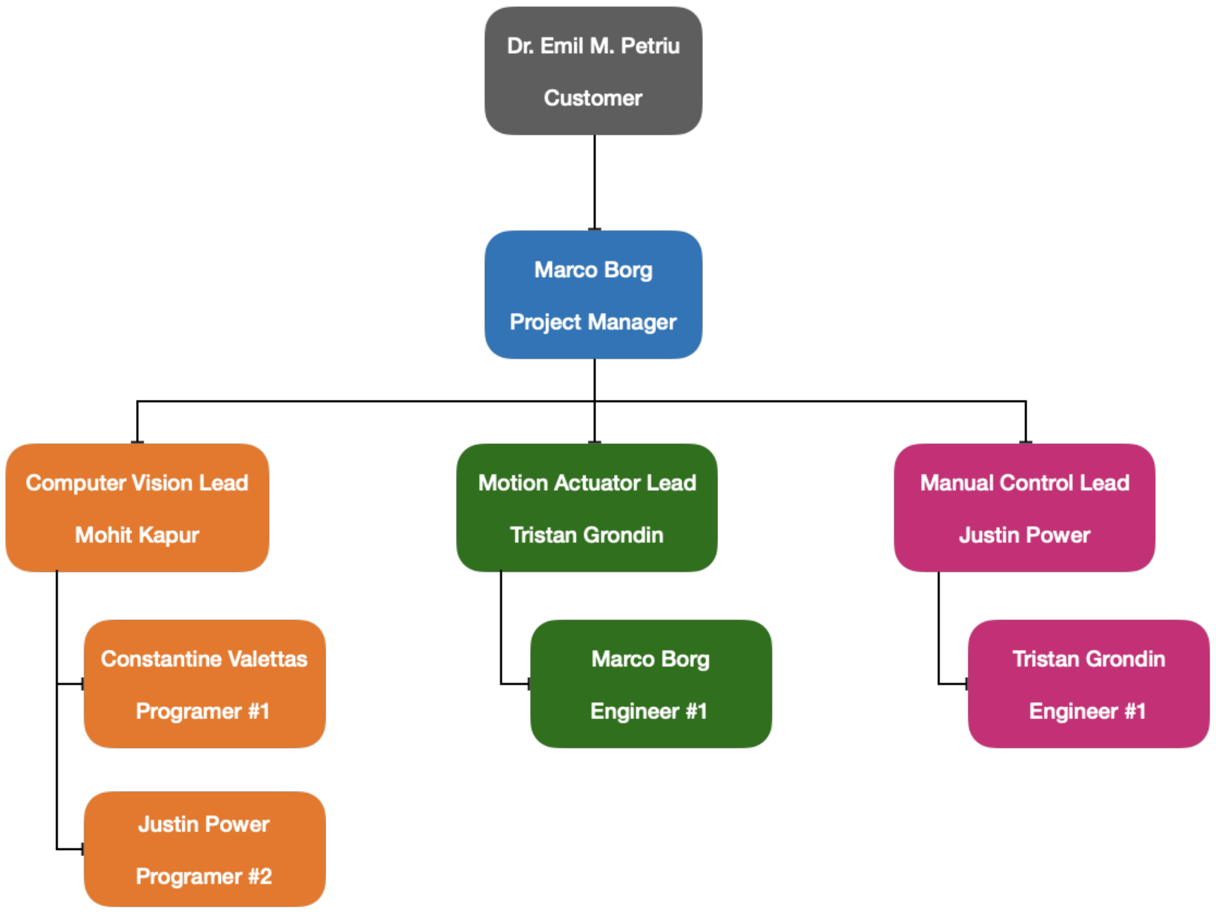


Figure 1

3. Software Design Description

The software design description was created to show how each sub-component functions and operates on a micro-scale. Therefore, the following descriptions will show what each component in the software will look to accomplish as well as indicate how the code should be constructed. This will document will look to serve the software developers assigned to this project to help guide the development process for each sub-component. The following figure will show how the project structure will look and designate the corresponding sub-components to their respective design lead. This figure was obtained from the project charter of this report which can be accessed in the references section of this report.



3.1. Main

3.1.1. Component Interfaces

This component will be responsible for beginning the program and initializing all parameters before the system will intake data from the user. Thus, the camera and actuators will be initialized so that when they are called to be utilized, all parameters will have already been set making computational speeds much faster. The input into this component of the code will come from the switch as well as the mode switch. There will be no outputs and will only serve to call upon other components when necessary while remaining in a loop to constantly search for new user input.

3.1.2. Component Design Description and Workflow

Figure 3 shows the package diagram for the main program routine. When the controller is turned on the main program is initialized and will begin scanning the switch on the main controller for which mode is currently active. The two different modes are automatic mode and manual mode. The main program will run in a continuous loop to scan for the active mode. If the switch on the controller is set to manual mode, the main routine will call the 'Call Button Sub-Routine' followed by the 'Call Actuator Subroutine K = 1,' if the routine detects the switch in automatic mode, the main program will call the 'Call Computer Vision Sub-Routine' followed by the 'Call Actuator Sub-Routine K = 0.' After either path is completed, the main program will return to the beginning and repeat the process.

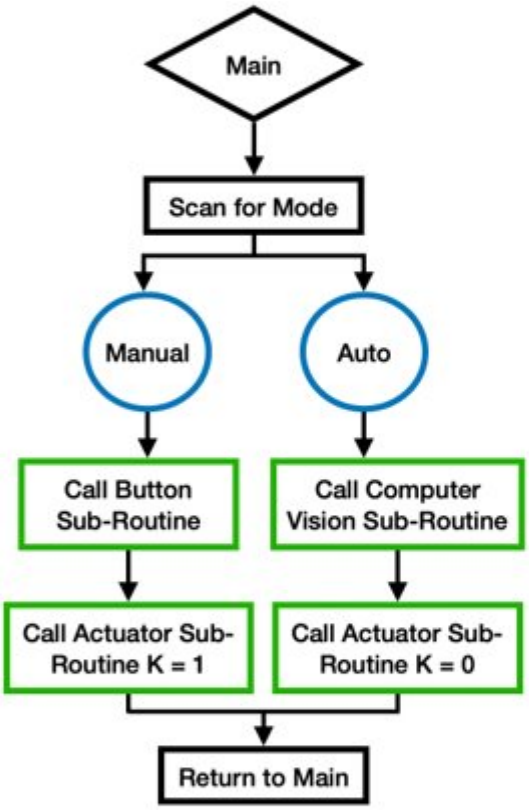

Main
Scan for Mode
Manual
Auto
Call Button Sub-Routine
Call Computer Vision Sub-Routine
Call Actuator Sub-Routine K = 1
Call Actuator Sub-Routine K = 0
Return to Main

Figure 3

3.1.3. Software Requirements Mapping

This component is responsible for detecting whether the system is in automatic mode or manual mode according to the SRS.

3.2. Call Button Sub-Routine

3.2.1. Component interfaces

The Call Button Sub-Routine (Figure 4) will interface with the buttons connected to the microcontroller. It will take a 1 or 0, where 1 represents the Up button being pushed, and 0 represents the Down button being pushed. The corresponding input will also be the output of the routine that is sent to the Call Actuator Sub-Routine K = 1.

3.2.2. Component Design Description and Workflow

The Button Sub-Routine starts when the control switch is in manual mode and is called by the main routine. The routine begins by scanning the controller for any inputs from the up or down buttons. If nothing is detected, no output is sent and the subroutine will continue to scan for inputs from the buttons until there is an input, or the main routine instructs the sub-routine to stop. If an up or down signal is detected, then the input will also act as an output and send the respective button signal to the Actuator Sub-Routine.

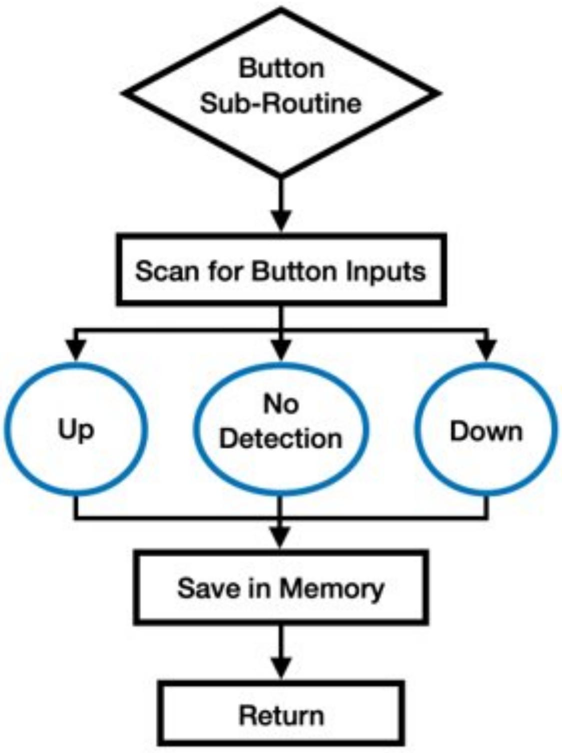


Figure 4

3.2.3. Software Requirements Mapping

This sub-routine fulfills the following SRS requirements:

REQ-1: The 2 buttons must send their own signal to the microcontroller when pressed and held. [R1]

REQ-2: The buttons must stop sending their respective signal when the button is released by the user. [R1]

REQ-3: The microcontroller must be able to identify if the up or down button is being pressed and held. [R1]

REQ-4: The microcontroller must ignore all inputs if both the up and down buttons are being pressed simultaneously. [R1]

3.3. Call Computer Vision Subroutine

3.3.1. Component interfaces

The Call Computer Vision Subroutine will interface directly with a Raspberry Pi Microcontroller.

3.3.2. Component Design Description and Algorithms

The Computer Vision Subroutine (Figure 5) begins when the control switch is in automatic mode and is called by the main routine. The routine begins by scanning for a user's position and determining if the user has changed positions. If the users detected position remains within set bounds, the system will not change its position. The system will continue to check for a change in position until the user moves, or the routine is aborted. If the subroutine detects a change in position, it will send a message to the actuators sub-routine of which direction the user is moving in.

The Computer Vision Subroutine records a user’s position using cartesian coordinates x and y, with x describing the horizontal axis of the captured video, and y being the vertical axis. The subroutine analyzes the y-coordinate values of the detected user and records the average y-value in pre-specified intervals. The average y-position of the current interval is compared to the average y-position of the previous interval to determine which direction the user is moving. This algorithm continually runs until a stop command is issued from the system.

The Haar Cascade Model used will be able to detect if a face is present in frame by performing a series of tests, each increasing in specificity. The cascade model creates a rectangular window that scans the live video/photo from left to right, and top to bottom analyzing for features. Due to the natural lighting of faces, the separate shapes are split into darker and lighter segments. The features themselves are composed of a value obtained by taking the difference of the sum of dark pixel segments from the sum of pixels in the light segment. If a predetermined feature is detected, the scanner performs another scan. This repeats until all stages are passed by the window, displaying the face region.

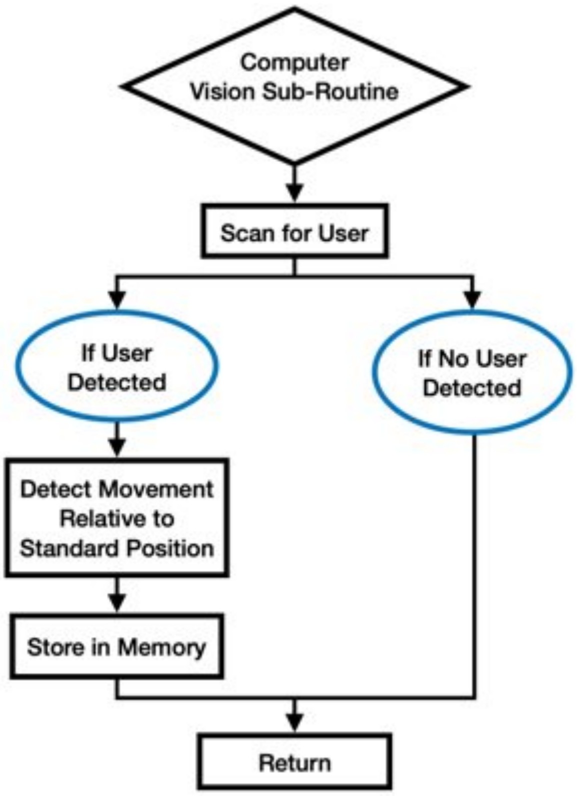

Computer Vision Sub-Routine
Scan for User
If User Detected
If No User Detected
Detect Movement Relative to Standard Position
Store in Memory
Return

Figure 5

The initial goal was to have each sub-routine running from the Raspberry Pi. During the process of setting up the Pi to be used, there were a series of issues discovered. The main issue was with the camera itself. While testing the camera, the resolution was found to be extremely poor meaning it would not be able to work as intended. Along with this, there were a series of issues when it came to installing OpenCV on the Pi itself. To work around this, a socket was used. For the prototype designed, the CV script was running off a separate device. The socket was implemented to allow for communication between the device running the CV and the Pi running the actuator software. On the Pi, a server written in C was designed to receive instructions from the Python CV, acting as the client. From here, the actuators are activated as necessary.

3.3.3. Software Requirements Mapping

REQ-1: The computer vision camera must be able to detect the head and shoulders of the user. [R1]

REQ-2: The computer vision camera must be able to detect the vertical position of the user's shoulders relative to the workspace's surface. [R1]

REQ-3: The computer vision camera must only detect movement from the user directly in front of the camera and not any person behind the user. [R1]

REQ-4: The Computer vision camera must indicate to the microcontroller if the user is rising or declining in height. [R1]

REQ-5: The Computer vision must send output via a Raspberry Pi socket to the main actuator program. [R1

3.4. Actuator Sub-Routine

3.4.1. Component Interfaces

This component (Figure 6) will begin when it is called by the main program. It is responsible for interfacing with the two linear actuators of the system. There are 2 workflows for this subroutine which are dependent on the mode detected by the switch. If manual mode is detected it will instruct the actuators to move in the direction according to the signal being read from memory (Up or down). If the automatic mode is detected it will move in the direction that is stored in memory from the Call Computer Vision Subroutine.

3.4.2. Component Design Description and Workflow

The first step of the Actuator Subroutine is to read from memory which mode the system is in. If the controller is in manual mode, it will read the memory for the signal that is stored by pushing the buttons. If the signal stored in memory is the up signal, then the routine will instruct the actuators to move up by turning on the negative polarity relays connected to the actuators, allowing them to move up. If the signal stored in memory is the down signal, then the positive polarity relays are turned on, allowing the actuators to move down.

However, if the controller is set to Automatic mode, it will read from memory which signal is being sent from the detection script. If the script detects the user moving up, it will send a signal to the actuators to raise the height of the system using the negative polarity relays connected to the actuators themselves. If the script detects a decrease in user height, then a signal will be sent to activate the positive relays to move the system down.

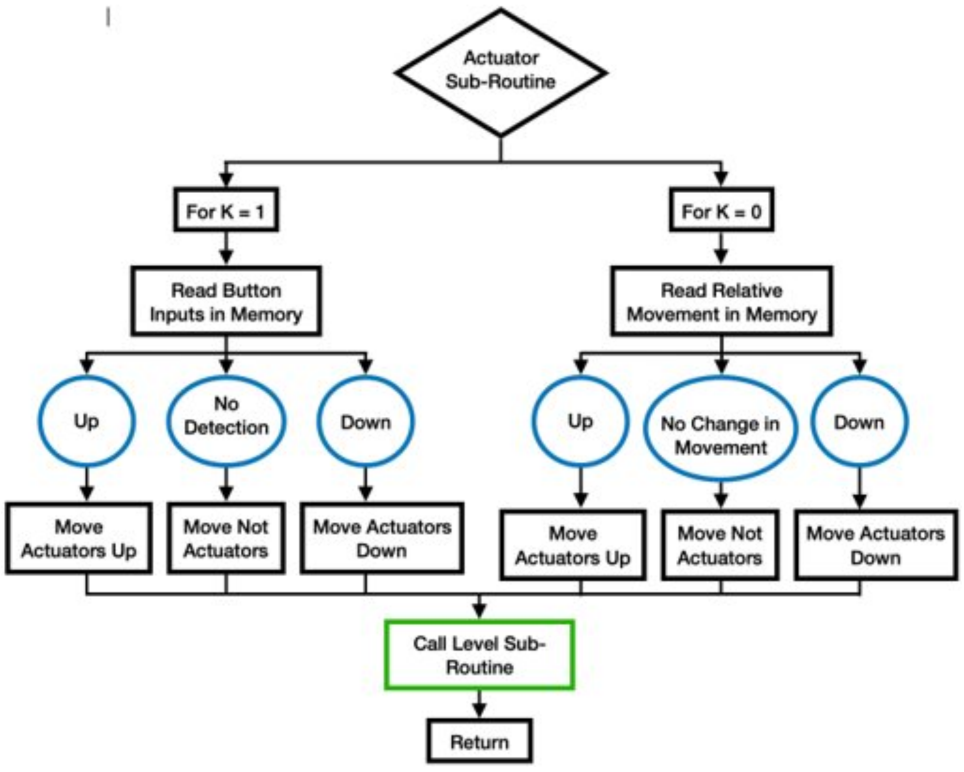


Figure 6

Figure 1: Main Routine

3.4.3. Software Requirements Mapping

REQ-5: The microcontroller must communicate to the actuators to move up if the up button is being held or move down if the down button is being held. [R1]

REQ-5: The microcontroller must instruct the actuators to move up if the user is ascending and move down if the user is descending. [R1]

REQ-6: The actuators must move down if the user is declining and move up if the user is ascending. [R1]

3.5. Level Sub-Routine

3.5.1. Component Interfaces

This routine will interface with the actuators and IR distance sensors via the raspberry pi microcontroller. It will be responsible for always keeping both actuators at the same level. This is needed in case the actuators have different movement speeds. Since this will be used on a workstation, the actuators need to always have the same vertical position to keep the workspace level.

3.5.2. Component Design Description and Workflow

The Level Sub-Routine (Figure 7) is called when the actuators are instructed to change positions via the Actuator Sub-Routine. It will begin reading the position of each actuator using the IR distance sensors relative to the minimum position of each actuator. The 2 positions are repeatedly stored in memory and compared to check for equality. Since the goal of this Sub-Routine is to keep both actuators at the same height, there will be no action performed by the Level Sub-Routine if the 2 actuators move at the same rate while also having the same position in the vertical dimension.

However, if the routine detects an inequality in the position of the actuators it will run through a series of steps to realign the 2 actuators back to the same position. First, it will read from memory which direction the actuators were instructed to move in. Second, it will read from memory which actuator is in a lower position relative to their rest positions.

If the Left Actuator is lower than the right, the following will occur:

1. Read from memory if the desk is moving up or down.
    - If the desk is moving downwards the left actuator will stop, and the right actuator will continue moving until the routine reads that both actuators have the same position.
    - If the desk is moving upwards the right actuator will stop, and the left actuator will continue moving until the routine reads that both actuators have the same position.
2. Both actuators will continue moving until they are instructed to stop or there is another inequality in their relative positions.

If the right actuator is lower than the left, the following will occur:

1. Read from memory if the desk is moving up or down.
    - If the desk is moving downwards the right actuator will stop, and the left actuator will continue moving until the routine reads that both actuators have the same position.
    - If the desk is moving upwards the left actuator will stop, and the right actuator will continue moving until the routine reads that both actuators have the same position.
2. Both actuators will continue moving until they are instructed to stop or there is another inequality in their relative positions.

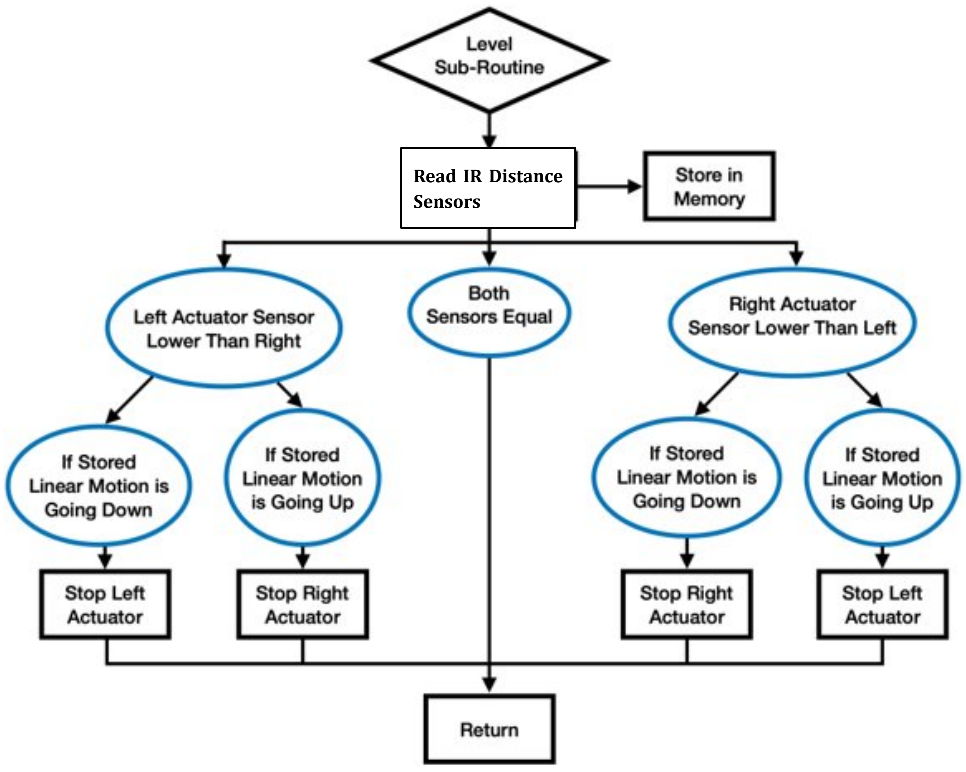

Level Sub-Routine
Read IR Distance Sensors
Store in Memory
Left Actuator Sensor Lower Than Right
Both Sensors Equal
Right Actuator Sensor Lower Than Left
If Stored Linear Motion is Going Down
If Stored Linear Motion is Going Up
If Stored Linear Motion is Going Down
If Stored Linear Motion is Going Up
Stop Left Actuator
Stop Right Actuator
Stop Right Actuator
Stop Left Actuator
Return

Figure 7

3.5.3. Software Requirements Mapping

REQ-1: The IR distance sensors must be able to record and send the position of their respective actuators to the microcontroller. [R1]

REQ-2: The microcontroller must be able to identify which actuator is in the wrong position. (If the desk is moving up then the lower actuator is in the wrong position and if the desk is moving down then the higher actuator is in the wrong position). [R1]

REQ-3: The microcontroller must be able to adjust the height of only one actuator at a time and both actuators at the same time. [R1]

REQ-6: The actuators must move down at the same rate if the microcontroller indicates a down input and move up at the same rate if the microcontroller indicates an up input. [R1]

REQ-7: Both actuators must be always at the same height. [R1]

4. COTS – SOUP identification

The only external library being used in the development of the Body Tracking Actuator System is OpenCV. OpenCV is a free-to-use, open-source python library containing the required tools to build a body tracking system.

5. References

Cascade classifier. OpenCV. (n.d.). Retrieved April 9, 2023, from https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html