San Diego State University

Robotic Arm Controller Manual

Austin Owens

May 1st, 2016

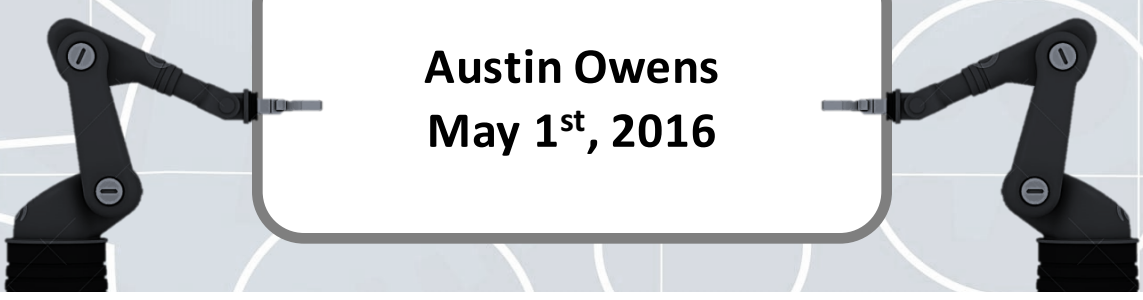

Table of Contents

Launch GUI Module

_launch_gui_ is the module that is run to initiate the graphic user interface (GUI) as seen in Figure 1. This module is responsible for almost all of the aesthetics/widgets that are seen on the GUI as well as updating the program when the widgets are controlled by user input. All code in this software is called either directly or indirectly by the ***_launch_gui_*** module. In the module, there is a class called ***GuiCreation*** that contains two functions, ***updateProgram*** and ***guiSetup,*** which are designated areas for code responsible for updating the whole program and the initial setup of GUI widgets. The code that is in the ***updateProgram*** function will be iterated through several times and contains everything that is happening behind the scenes until the program terminates. The code in the ***guiSetup*** function will be iterated through only once to setup all the widgets in the GUI.

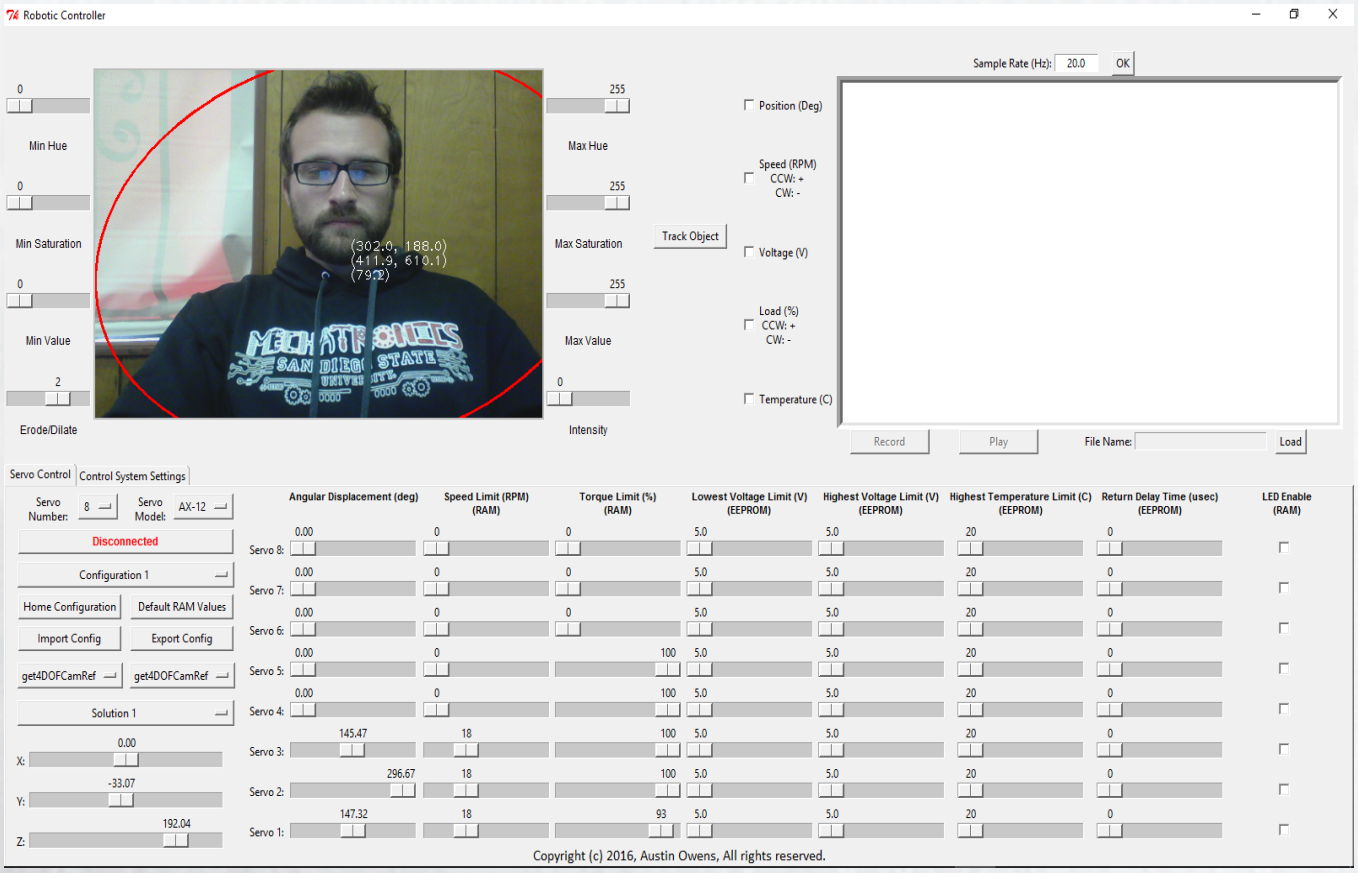

Figure 1

Callback Functions

Oftentimes when certain widgets are set up, they require a dedicated function to call when the widget is activated, such as a button push. These type of widgets that require a function to be written and executed when the widget is activated is called a callback widget. The code for all callback widgets are located in the ***callback_functions*** module.

Event Handlers

An event handler is a callback routine that operates asynchronously and handles inputs received from either key strokes or mouse activity when the GUI is running. The module ***event_handlers*** contains all the code specific to event handlers, such as pressing drawing a box on the raw image feed with the mouse to select a region of interest to track with image processing, as seen in Figure 2.

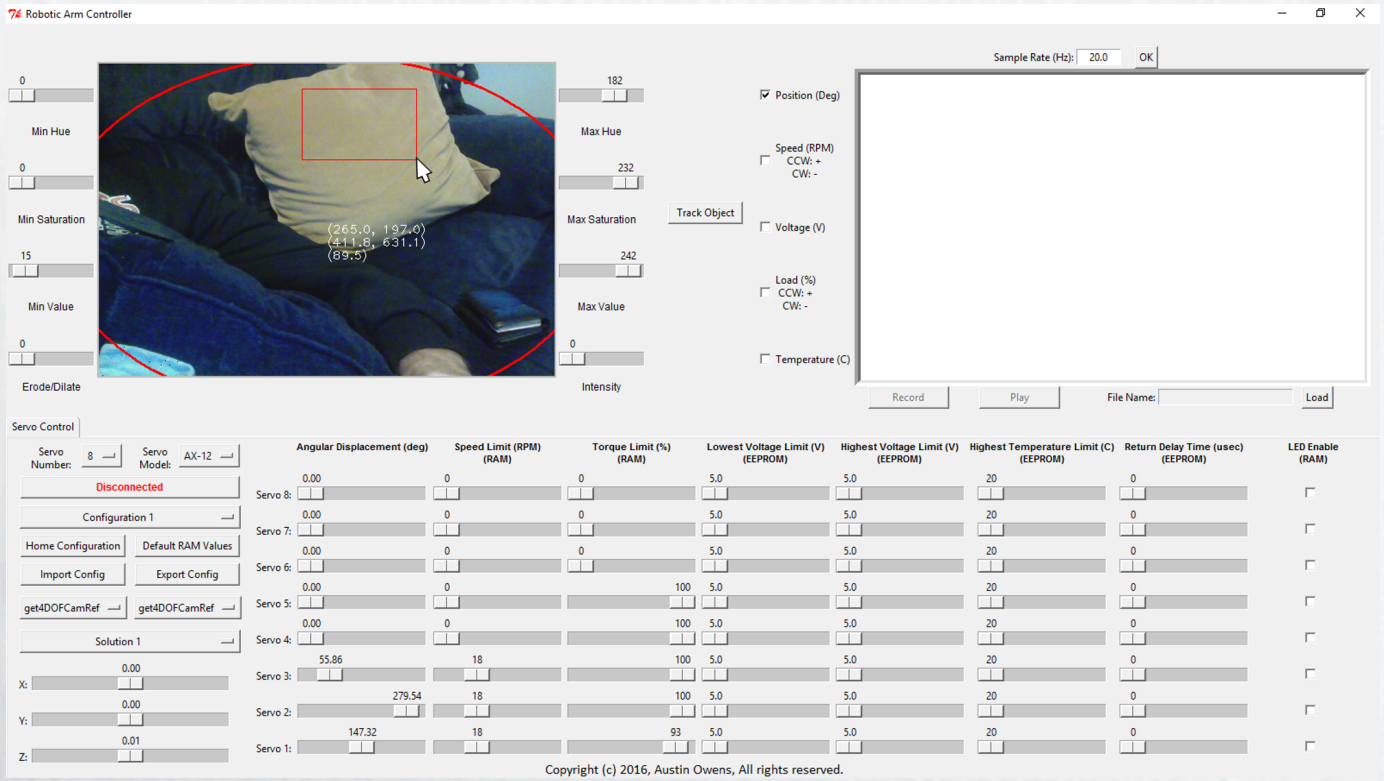

Figure 2

Image Processing

The ***image_processing*** module handles all image processing and object detection that's taking place in the software. The image processing code is generic to fit most tracking needs from an image processing and object detection perspective and is by no means capable of solving detecting all objects in all types of environments. In order to have image processing code suited towards a specific applications, the current code in ***image_processing*** is excellent boiler plate code to start off with. The image processing filters used within this module are Hue, Saturation, and Value (HSV), and Erode/Dilation. The object detection algorithm used within this module is called Continually Adaptive Meanshift (CAMshift). Examples of the image processing capabilities are shown in Figures 3 and 4.

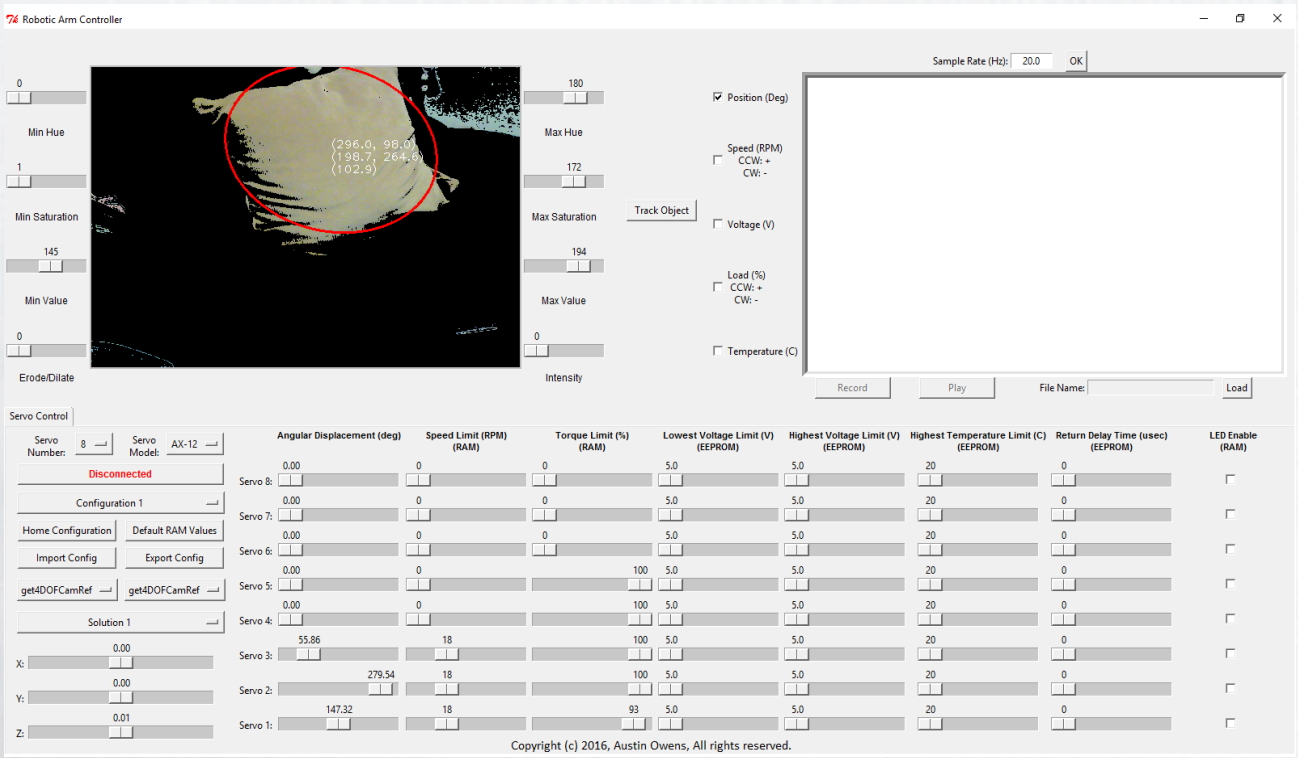

Figure 3

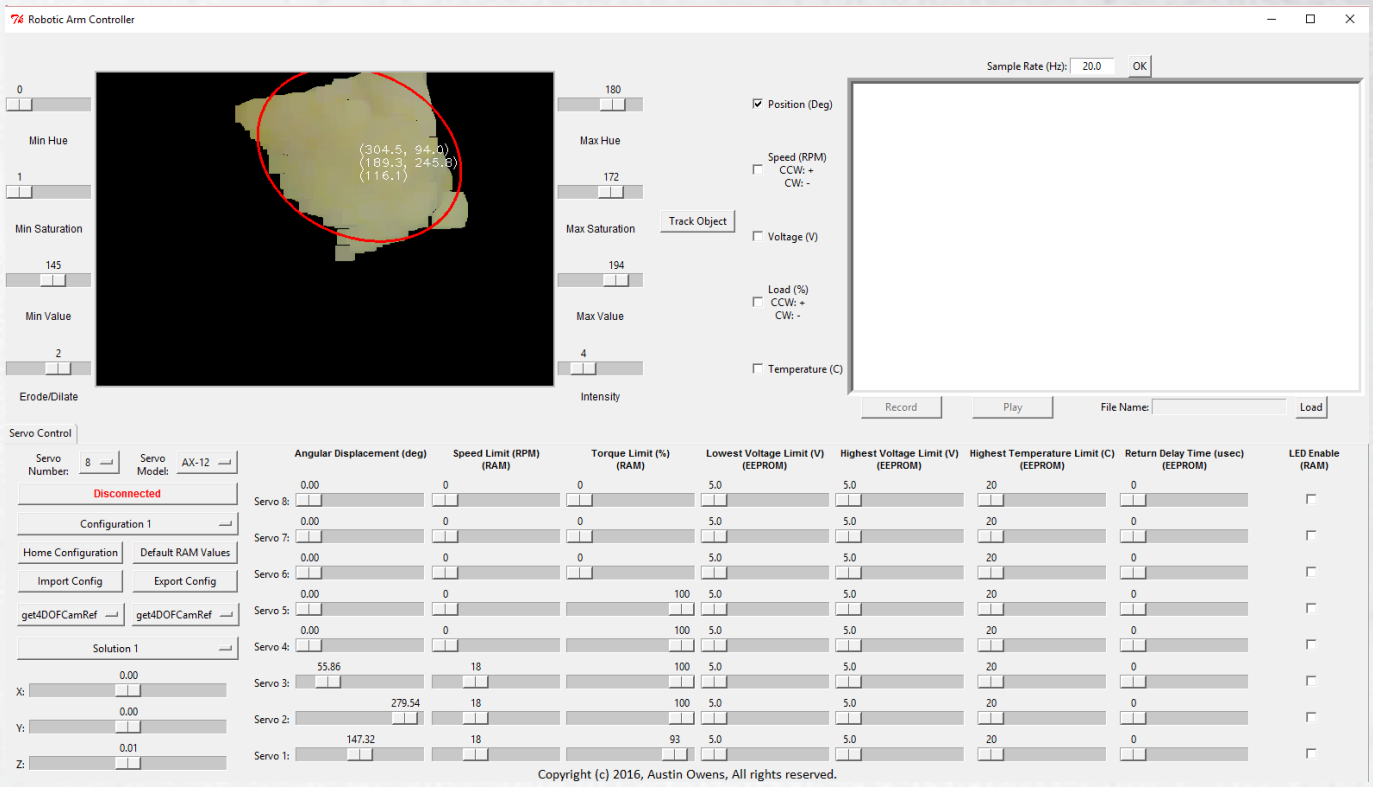

Figure 4

IP Movement Algorithms

IP_movement_algorithms contains algorithms in how a robotic arm should move from image processing input. Algorithms may only be written for robots whose inverse kinematics is known and implemented into the software.

Kinematics

The *kinematics* module contains the forward and inverse kinematics to control serial linkage robots. The forward and inverse kinematics is most likely different for every serial linkage robot. As a result, all the user needs to do is follow the instructions in the module in how to create the functions in a specific way and the software will automatically parse the functions and incorporate it where necessary. If the user wrote the functions correctly, they will see their functions show up in the kinematic selector in the GUI.

Object Detection

The ***object_detection*** module is where the CAMshift code resides, it is being called by the ***image_processing*** module. CAMshift is ideal for tracking objects in images of all shapes and sizes. It also works very well in differentiating between noise and the object of interest.

Previous State Logging System

Reads and writes values to a text file so that next time the software runs, it will remember previous users input.

Utilities

The utilities module contains generic useful code, such as the ability to keep time throughout certain points in the code.

Dynamixel Comm Package

The ***dynamixel_comm*** package contains the ***dynamixel***, ***ax_series***, and ***mx_series*** modules. These modules are the lower level code that communicates to the Dynamixel servos. The ***dynamixel*** module is a generic module that gets inherited by the ***ax_series*** and ***mx_series*** modules. These modules are responsible for sending/receiving data packets to/from the microcontrollers on the Dynamixel servos.

The communication protocol is in half-duplex UART and a specific data packet structure must be formed to communicate to the Dynamixels. To find out more information on the data packet protocols and structure of the communication, a simple google search of the Dynamixel model will yield a datasheet that goes with the servo.