The first step in my marker detection scheme is calibrating the camera module. To do this, I run a function that sets the shutter speed, the iso, and finds the optimal white balance gain of the camera. A calibration picture is taken within this function using the camera.capture() function. The image is stored in the file path and can be used to judge whether the settings and values of the camera module are optimal.

The second step in my marker detection scheme is the function that takes a picture using the camera module, detects whether a marker is in the image, performs an analysis on the corners of the detected marker, and then finally determines which quadrant the marker is within the image. The function takes in as argument the predetermine white balance gain of the camera. Within in the function the camera gain is set this value and then a picture is taken. This is to ensure that an optimal picture is taken, such that any markers in the frame of the camera will be detected. After a picture is taken, an image object is created from the taken picture. This image object is then converted to gray scale. Then the aruco dictionary and parameters are set. The aruco detection function is then ran using the gray scale image, aruco dictionary, and aruco parameters as function arguments. The associated corners, ids, and rejected image points of a detected aruco marker are stored in variables from running the aruco detection function. An if conditional statement within the function checks whether at least one id of an aruco marker is detected. If there is not at least one id detected, the function will return the value four and break. The value four will make sense for obvious reasons discussed later. However, if a marker id is detected, the stored marker corners will be used to determine the pixel x and y coordinates of the center of the marker in the image. These center x and y pixel coordinates of the marker will be compared within an if-elif-else conditional branch to determine which of the four possible quadrants the marker is in. If the marker is in the first quadrant (0 deg – 90 deg) the function will return a zero. If the marker is in the second quadrant (90 deg – 180 deg) the function will return a one. If the marker is in the third quadrant (180 deg – 270 deg) the function will return a two. If the marker is in the fourth quadrant (270 deg – 360 deg) the function will return a three. Finally, if no marker is detected or the marker center manages to be at the intersection point of the quadrants, the function will return a four. Each returned value will be sent to the Arduino, which will assign the correct angle from the returned value that the motor needs to rotate to.