**Core Programming Module Requirements**

**Projects: Controller for Portable Wide-Field Scanning Aberrometer**

Author: Hope Queener

Developers: Hope Queener, S. Harsha Tata

Principal Investigators: Geunyoung Yoon

Created: 1/10/2022; Updated 4/22/2022

# Introduction

This requirements document describes software control of a custom, portable, wide-field, scanning aberrometer (wavefront sensor) system in the laboratory of Geunyoung Yoon. Lab member Chloe Degre Kendrick is the key person involved in the system design. The system captures Hartmann-Shack wavefront sensor (WFS) spot images while the image is being translated horizontally along a rotational axis that pivots at the subject’s pupil. The WFS camera field of view can be translated vertically with an additional motor, or the two can by synchronized to provide diagonal movement. A third motor is used to adjust the plane of the camera along the optical axis for best focus position. A fourth motor adjusts the tilt of the plane of the beam-splitter so that the plane of the image is parallel to the plane of the camera. The system also requires a pupil camera for alignment. The software provides controls for setting the scanning directions, rate and timing for capturing WFS images.

Note that this document describes the long-term goals for this system. The project will progress in phases that are described in the “Development Phases” section, starting with only the horizontal scanning motor and the WFS and Pupil cameras.

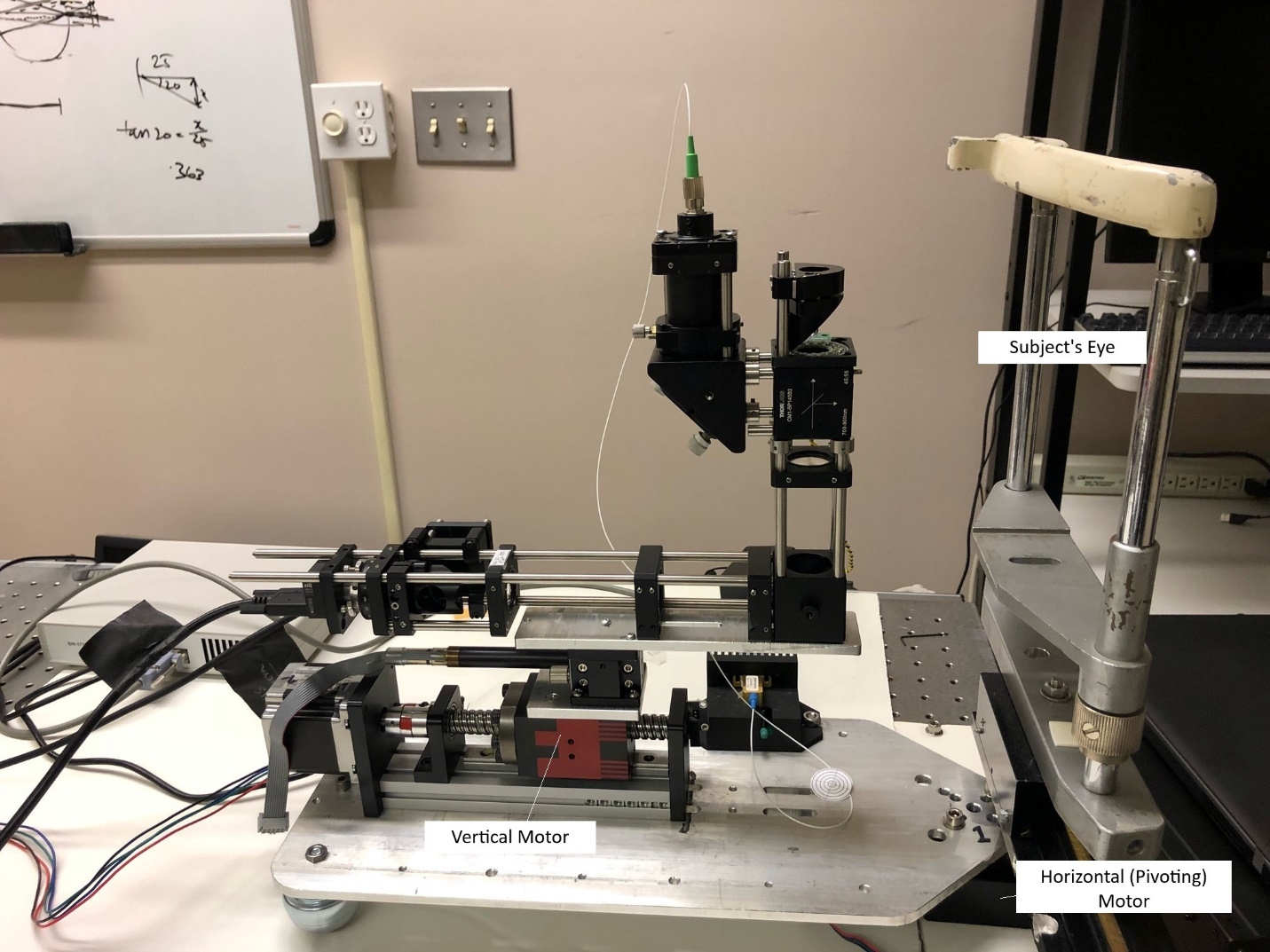


Figure 1. The Scanning system

Figure 1 shows the scanning system (photo taken January 1, 2022). The subject views the system from the headrest. The Horizontal (pivoting) motor moves the system on an arc along the temporal-nasal axis. The pivots point centered below the center of subject’s pupil. The Vertical (superior-inferior) directions are relative to the plane of the subject’s pupil, taking in to account the beam-splitter in the system. The vertical motor translates parallel to the optical bench. The optical axis is oriented vertically to the optical bench. The optical axis motor moves the stage up and down to set the optical distance from the pupil plane.

# Components

## Wavefront Sensor Camera

Ximea model MQ013RG-E2 is a near infrared monochrome camera used to capture the Shack-Hartmann spot image used to measure the wavefront.

## Pupil Camera

Ximea model MQ013MG-E2 is a monochrome camera used to view the pupil. When the image of the pupil is well-centered, the WFS optical system is also well-aligned.

## Stepper Motors

Three of the stepper motor drivers are STEPPERONLINE model DM542T. These drive the horizontal, vertical and optical axis motors.

<https://www.omc-stepperonline.com/digital-stepper-driver-10-42a-20-50vdc-for-nema-17-23-24-stepper-motor-dm542t.html>

A fourth stepper motor controls the tilt of the beam-splitter. Its driver is the Thorlabs Kinesis® Brushed Motor Controller model KDC101. The software control options for this driver include the Kinesis® software package (.NET or low-level DLLs) or a legacy, ActiveX APT package. Since ActiveX technology is legacy technology, the Kinesis® package will be implemented.

<https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=2419>

The motor model numbers are:

Motor 1: BaQian

Motor 2:

FUYU FSL-40E5010C7

# 50mm Stroke CNC Ball Screw Stage Linear Guide Actuator Motion Slide Rail G1610 Ballscrew Nema23 Motorized Stepper Motor.

The compatible stepper motor driver looks the same as the DM542T.

<https://www.aliexpress.com/item/32589549569.html?spm=a2g0o.store_pc_home.productList_6001938431133.subject_1>

The drawing for this stepper motor indicate a ‘step angle” of 18 degrees.

Motor 3: (Tilt)

17HS08-1004S

1.8 degree step angle

Thorlabs Inc. Z625B

25 mm Motorized Actuator with 3/8" Barrel

This product has been replaced by the Z825B. It allows very small step sizes over the entire travel range.

The KST101 stepper motor driver gives a theoretical travel per microstep of 0.5 nm.

DIP switches on the drivers determine the steps per revolution.

|  |  |  |
| --- | --- | --- |
| Motor | Steps/Revolution | DIP switches |
| 1 | 4000 | ON-OFF-ON-OFF |
| 2 | 4000 | ON-OFF-ON-OFF |
| 3 | 4000 | ON-OFF-ON-OFF |

To be added:

Arduino processor

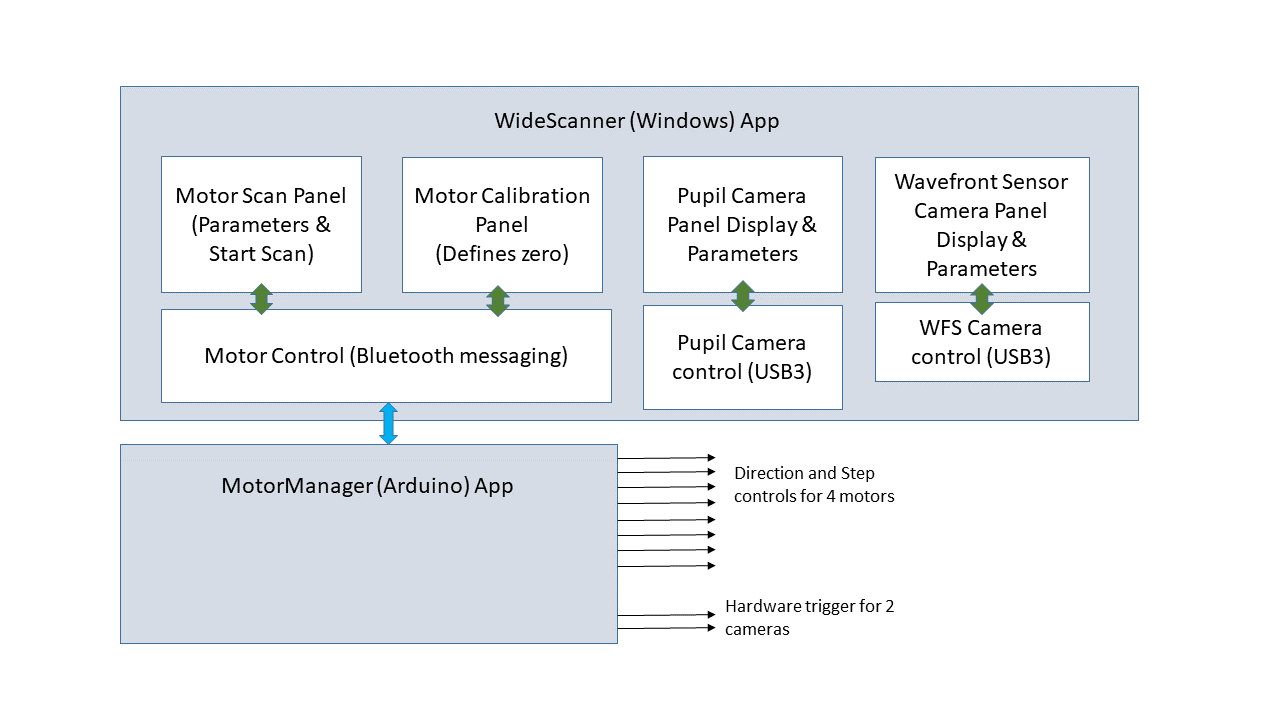
Bluetooth HC-50

# Software Development Environment

The project leverages two existing projects that were previously developed for the Yoon lab. CorneaWFScanner, was developed by Keith Parkins at the University of Rochester to provide pupil and wavefront sensing image capture. CorneaWFScanner uses Qt for the user interface (UI) and C++ code to manage camera control and send serial commands to 2 Velmex motor controllers for a different rotational stage system. The second project is an Arduino application that can control the Stepperone motors used for this system based on analog velocity controllers (knobs that control potentiometers) and digital direction and start/stop signals. The Arduino logic was coded by Marty Gira at the University of Rochester. The file is named: Arduino\_Stepper\_control\_final.ino.

The code for the Wide-Field Scanning Aberrometer is comprised of a Windows Qt/C++ program and an Arduino program. The Qt/C++ program provides the UI for controlling the cameras and motors, the operational code for the cameras, and messaging code for the motors. The application name is “WideScanner”, for brevity. Much of this code can re-use the CorneaWFScanner project. A new UI panel for the Stepperone motors replaces the UI panel for the Velmex motors. The new Arduino code handles the operation of the motors based on parameters sent by the Windows application (instead of by hardware controls). Communication between the host and the Arduino is performed via BlueTooth. (If Bluetooth communication is not sufficient for the timing needs, hard-wired serial communication may be employed instead).

Existing Arduino code that is associated with the hardware motor controller defines which Arduino output pins send step pulses to the motors. The Arduino will continue to reside in the existing hardware motor controller. The hardware controls may or may not be available when running the aberrometer control, depending on how much of a delay it adds to attend to both hardware inputs and software inputs while operating the scanning loop.



For the best possible camera synchronization, hardware triggers are provided to the Ximea cameras. As a first phase of image capture, the cameras can acquire continuously during the sweep or at times managed by less precise software triggers.

# Parameters

The Windows application provides for camera control and motor control.

## Motor Calibration Panel

The UI provides a Calibration module which moves each motor independently in coarse or fine steps in the requested position. This process is used to define a state where the motor is at the centered (zero) position for all motors. At the end of every sweep operation, the software returns the motors to the zero position. The optical axis is set a “relative” zero position indicating best focus.

The Calibration UI panel provides these independent controls for the 4 motors.

* Horizontal (nasal-temporal) Motor
* Vertical (superior-inferior) Motor
* Tilt Motor
* Optical Axis

The UI parameters are described in the table below. Note that the translation of one UI step is defined by the UI and is distinct from the translation of the stepper motor steps.

| **Parameter** | **Units** | **Type** | **Operation/Status** |
| --- | --- | --- | --- |
| Minimum Position | Degrees | edited text | Negative value that defines the most negative value permitted by the software. This limit is defined relative to the user-defined zero position. |
| Maximum Position | Degrees | edited text | Positive value that defines the most positive value permitted by the software. This limit is defined relative to the user-defined zero position. |
| Enable Motor | True/False | checkbox | Motor moves only when checked |
| Coarse step size | Degrees | List-box | Defines the translation of one UI step in visual angle degrees in “Coarse” mode |
| Fine step size | Degrees | Label or list-box\* | Defines the translation of one UI step in visual angle degrees in “Fine” mode |
| Use coarse steps | True/False | Checkbox | Selects to use coarse steps when checked. Use fine steps when unchecked. |
| Step Backward | n/a | Pushbutton | Take one UI step in minus degree direction. Button is disabled until the step is completed to prevent excessive repetitions by the user. |
| Step Forward | n/a | Pushbutton | Take one UI step in plus degree direction. Button is disabled until the step is completed to prevent excessive repetitions by the user. |
| Move Backward | n/a | Pushbutton | Take continuous steps in the minus degree direction until the user presses the “Stop” or “Stop All” button on the UI, or the motor reaches the limits defined by the Minimum Position parameter. |
| Move Forward | n/a | Pushbutton | Take continuous steps in the plus degree direction until the user presses the “Stop” or “Stop All” button on the UI, or the motor reaches the limits defined by the Maximum Position parameter. |
| Stop | n/a | Pushbutton | Stops the respective motor. |
| Stop All | n/a | Pushbutton | Stops all motors. |
| Zero | n/a | Pushbutton | Return to zero position |

\*The Fine Step Size may simply be a label that indicates the degrees per pulse step provided by the stepper motor controller, for the user’s reference. If these steps are too small, the fine step could be a list box that provides for a limited number of options for 1, 2 or 3 pulse steps and the equivalent degrees.

The host computer informs the Arduino when the motors are in the “zero” position of their respective axes, and then the Arduino internally maintains each motor’s position as the number of steps from zero at all times, so that it can return to zero.

## Sweep/Capture Panel

The Sweep panel provides for these text editing controls for the four motors and step locations for image capture:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Starting Position | Step | Ending Position | Duration of Scan |
| Units | Degrees | Degrees | Degrees | seconds |
| Horizontal (nasal-temporal) Motor |  |  |  |  |
| Vertical (superior-inferior) Motor |  |  |  |  |
| Tilt Motor |  |  |  |  |

The arrangement of the panel will also allow for the horizontal and vertical motors to be treated as a single diagonal motion. (Refer to CorneaWFScanner for an example UI.) The tilt motor may be determined computationally based on the horizontal and vertical motors. However manual control will be enabled also. The optical axis motor generally is kept at a static distance during the sweep. However, it may be offset from the zero position for the sweep.

The step position determines when the pupil and wavefront sensor cameras will capture the image. For the initial phases of the project, these cameras will capture continuously or in bursts during the sweep. At later stages, the cameras are set up to be triggered by a subsequent hardware pulse. During the sweep, the Arduino issues a pulse at the appropriate step positions.

This panel has the following command buttons for the sweep operation.

| **Parameter** | **Operation/Status** |
| --- | --- |
| Initialize | Moves the motors from the zero position to the most negative position of the sweep. (The actual position may be adjusted for a calibrated “start-up” distance if the motor requires a short start-up interval for acceleration.) |
| Capture | Starts the sweep and image capture operation. If the motors are not in the initial position, the “Initialize” operation is automatically performed first. |
| Stop | Stops all motors. |
| Zero | Return to zero position |

## Camera Parameter Panel

This panel design is TBD based on the PupilCamera application and the CorneaWFSensor applications. Both of these previous programs use Qt for the UI, C++ for the internal logic. They provide for camera parameters and capture.

# Typical Sequence

The software returns all motors to the zero position when it closes, so that the system starts with the motors close to the zero position for the next session. At startup, the operator makes any manual adjustments with the WideScanner Calibration panel using the last zero position as a reference and sets the current motor position as the zero position for the alignment step.

Next, the operator aligns the subject in the head-rest. The pupil camera and wavefront sensor are put in “live” mode. The pupil camera is used to view the subject’s pupil as adjustments are being made. The wavefront camera spot image is also visual inspected for image quality. Aligning the subject may require additional Vertical (superior-inferior) and Optical-axis motor adjustments, head-position adjustments and seating adjustments. The Optical Axis Motor will generally be adjusted before the synchronized scanning mirrors operate, in order to set the best focal distance in the system centered on the pupil. The motor Calibration is re-set to zero to provide a suitable reference position for the sweep.

The operator next inputs the sweep parameters and confirms that the subject is in position. The user enters the positions of the synchronized motors and sets the “Initialize” command. When the initialize command is issued, the motors move to the starting position. Next, the user provides a “Capture” command. This command initiates the scanning and image capture sequence. The subject remains as still as possible during the sweep. Once the sweep is completed, the images are saved.

# Characterizing the Motor Timing

Members of the Yoon lab will characterize each motor’s range and timing ranges for typical scanning scenarios.

The general idea is to scan a single sweep (60 degrees of view) over 3 to 4 seconds. Timing of the WFS camera captures will be while the motor is scanning and will be impacted by the smear of the spots as the motor scans. However, the distortion is known and can be removed theoretically in post-processing.

The distance required for the motors to reach steady-state velocity and decelerate to a stop will need to be added to the starting and ending positions, respectively. The minimum and maximum acceptable velocity inputs will be provided by the lab based on preliminary experiments and the manufacturer specifications. The software will control motor rate and direction based on data provided by these characterizations.

# File Naming Conventions

The spot images are saved with the following naming conventions.

Will the operator be able to key in a base file name?

What image formats does the XIMEA camera support?

Can the timestamp (to the millisecond) be saved in image metadata or must an additional file be created?

One the images are captured by the Arduino processor they are copied to the host computer automatically. (Images could be renamed in this operation).

# Development Phases

Phase 1. The first phase of development is to control the horizontal (pivoting) motor and while capturing spot images from the WFS camera using the specified range, steps intervals and overall time interval. During phase 1, the Vertical motor is controlled by the custom, manual driver box which was built at University of Rochester. This driver provides buttons for stop/start and direction, and a knob for velocity.

Phase 2. To be added