Project at KUMC Precision Neural Dynamics Lab:

Object Track Matrix

Final Project Report

Author: Thresa Kelly

Date: May 11, 2023

Affiliates: Ryan Coppens, Dr. Alice Bean, & Dr. Adam Rouse.

Abstract

The goal of the object track matrix project is to design an activity board for a monkey reach-and-grasp experiment. The activity board consists of 12 objects connected to motion sensors and can be controlled using software. This project is overseen by the University of Kansas PHSX 601 professor Dr. Alice Bean and Neurosurgery Precision Neural Dynamics Lab lead Dr. Adam Rouse. The object track matrix will be designed by Thresa Kelly and Ryan Coppens. I will be responsible for the software design and experiment control systems. Ryan will design the motion sensors with wireless communication and construct the physical board.

1. Table of Contents

2. Background and Motivation

The University of Kansas (KU) Department of Neurosurgery Precision Neural Dynamics Lab, led by Dr. Adam Rouse, is interested in exploring how the brain controls the body. People with neurological diseases and injuries may have difficulty with brain-to-body communication. New innovations in technology and robotics can support people with paralysis and movement disorders [1]. The lab studies brain and spinal cord activity in animals and humans to better understand how groups of neurons coordinate arm reaching, hand grasping, and other movements. The lab also develops new computational methods to analyze the large datasets produced during experiments. The Precision Neural Dynamics Lab has four current research projects: neural encoding of precision movements, neural mapping of hand function, signal processing for epilepsy, and medical imaging algorithms [1].

The object track matrix project, which is described in this proposal, will facilitate reach-and-grasp experiments with rhesus monkeys. This aligns with the neural encoding of precision movements and neural mapping of hand function lab research projects. For the former project, the lab is interested in the target-independent and target-dependent neural signal limitations of speed and precision. Target-independent signals describe the neurons that fire for any movement for a given motor area. Target-dependent signals describe the level of activity of specific neurons associated with movement direction [1]. The latter project is interested in how the brain controls hand movement. The brain maps a specific hand grasp then sends the signal to the hand via the spinal cord; this can be studied by observing neurons for different categories of grasps and magnitudes.

The object track matrix is inspired by Dr. Rouse's 2015 paper: “Spatiotemporal distribution of location and object effects in reach-to-grasp kinematics” [2]. This study examined the movements of 22 joint angles from the shoulder to the fingers as the monkey performed reaching and grasping tasks as a function of time. The rhesus monkeys were trained to manipulate different objects on an activity board: the monkey would pull a coaxial cylinder and perpendicular cylinder against a spring, push a button, and rotate a sphere. An LED would illuminate the object that the monkey should grasp. Motion tracking, joint angle calculations, and temporal responses were recorded for each trial. The study found that reach-to-grasp neural control began with determining how to transport the arm and ended with the hand to grasp the object [2].

3. Description of the Project

The goal of this project is to design an activity board for a monkey to interact with and to develop software to control and record the experiment.

The activity board consists of a 4 x 3 grid of 12 objects each anchored to a rod. The activity board will lay flat on a horizontal plane. Each object has an LED that, when lighted, signals the monkey to pull the object. Each object will be connected to a wireless motion sensor; the accelerometer in the sensor will record how the object is moved and transmit data to the board's main processor. The objects on the rods will be interchangeable, which allows for several configurations. The objects will vary in size, shape, color, weight, and texture.

The software will allow the scientist to control the activity board and analyze the data. The user will be able to design the experiment by inputting how many trials to complete, the duration between trials, and how long the LED turns on. The software will also allow the user to design a specific object interaction sequence or to randomize the experiment. After setup is complete, the user can begin the experiment. The software will write the setup information and trial sequence to the object track matrix microprocessor. During the experiment, the software will read the current trial number and object moved and save this data to a file. After all trials are completed, the software will analyze the data and wave the success or failure status of each trial to a file.

For my contribution to the project, I have designed all of the software for the activity board control and the serial communication protocol between the software and hardware. I have developed the software from scratch using Qt, which is a cross-platform software for designing graphical user interfaces (GUIs), in C++ for Windows operating systems [3]. The software behavior has been discussed above. The software and hardware interface via a serial USB cable. The software converts commands into binary packets and writes to the activity boards processor. After sending a command, the software will wait for input from the board. My contributions are discussed in more detail in Section 5.

The relevant physical parameters for this project are timing at motion. The timing calculations from the hardware and software must be accurate to 10 ms. This ensures that the data from the activity board can be accurately matched to the concurrently measured neurological signals and body motion tracking. The largest constraint to timing accuracy will be from the continuous reading of wireless data from the accelerometer. We expect that the monkey will manipulate the objects for about 1000-2000 ms with an additional 250 ms added for moving the arm between resting and the object [2]. For motion, we are only required to track the object's motion in one dimension; this can be either pulling upwards, moving side to side, or twisting. The monkey must manipulate the object at least 1 cm to count as a successful interaction.

4. System Requirements

Physical constraints:

- Objects must be able to be disconnected from the activity board.
- Rods must be able to support objects of different size, shape, and weight.
- The activity board must support 12 objects in a 4 x 3 grid.
- The board must lay on a flat surface.
- The board should have an area of about 18 x 12 in.

Motion sensing:

- Accelerometer must be able to detect motion in one dimension.
- Noise from the accelerometer must not be interpreted as an interaction from the monkey.
- The object must be moved by 1 cm to be considered an interaction.
- The motion sensor package must be wirelessly attached to the end of the rod with the object.

Lighting:

- Only one LED should turn on at a time.
- The LED should turn on for a duration set by the user.

Software:

- The user must be able to plan an experiment by determining the LED sequence and timing durations.
- The software must read from the hardware in real time.
- The timing resolution of collected data must be accurate to 10 ms.
- The software must determine if a trial was successful (the correct object was manipulated) or a failure (no object or the incorrect object was manipulated).

5. Project Contribution

The project source code for the software is found on my GitHub [5]. The source code for the hardware is on Ryan's GitHub [6]. The control flow diagram for this project is shown in Figure 5.1. I will discuss my contributions to the project in the subsections below, including the serial communication protocol (Section 5.1), the software user interfaces (Section 5.2), the software internal design (Section 5.3), and the activity board microcontroller (Section 5.4). Ryan's discusses the activity board hardware and motion sensors in his report.

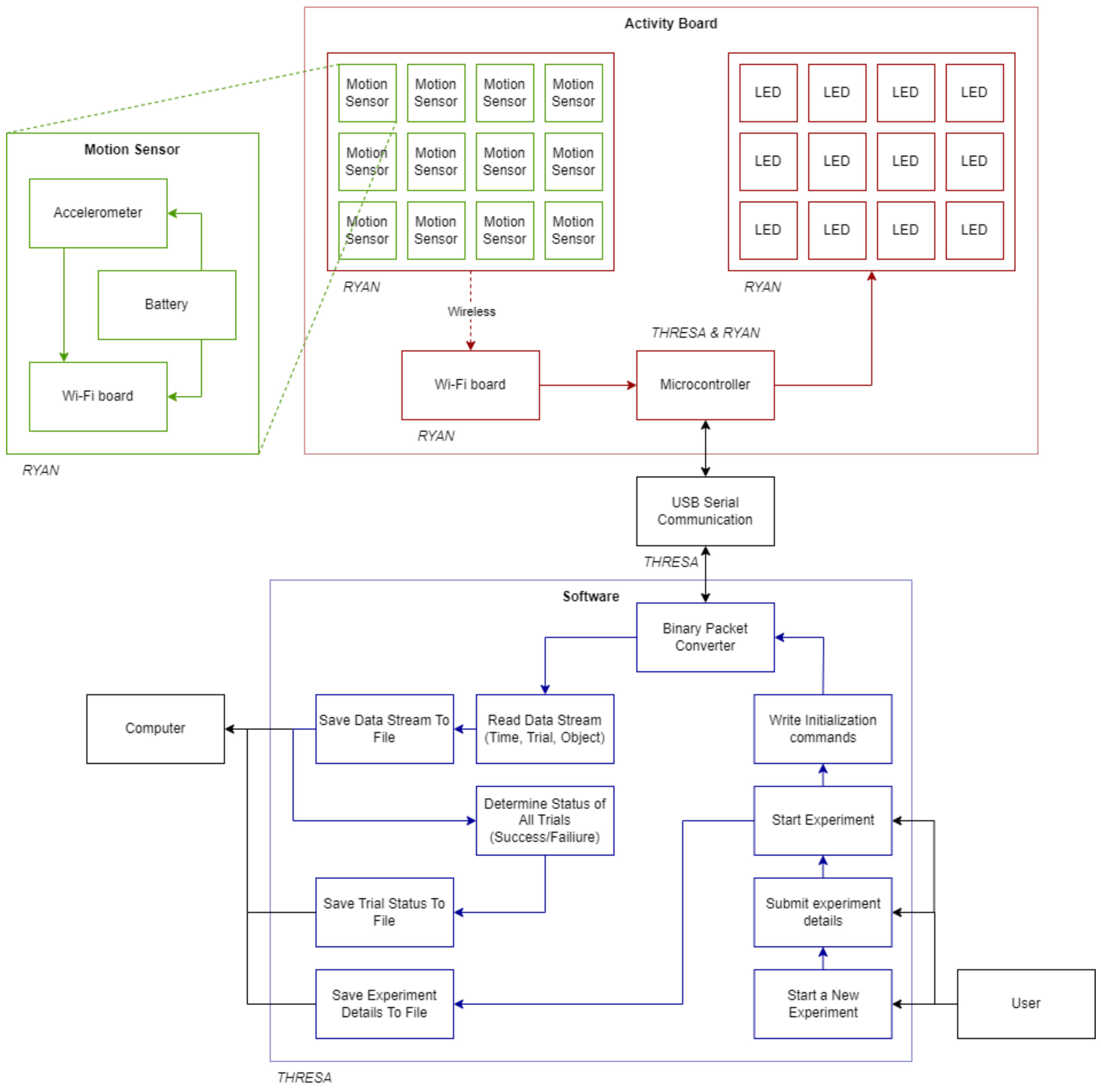


Figure 5.1. Object track matrix control flow diagram.

5.1 Serial Communication Protocol

The software and hardware communicate using a USB cable. I designed a serial communication protocol. The software will write command packets to the microcontroller, which will perform actions accordingly, then write relevant information back to the software.

Each serial write will contain a binary packet of 8 bytes, as shown in Table 5.1. The packet starts with STX (x02)[1], which flags that data is being sent. The next byte contains the command number, which can be 0-9. Next is 1 byte of the ID of the object moved, which can be 1-12 for an object or 0 for no object moved. Then there are 4 bytes of data; the contents are command specific. The packet ends with 1 byte of ETX (x03), to flag the end of the packet.

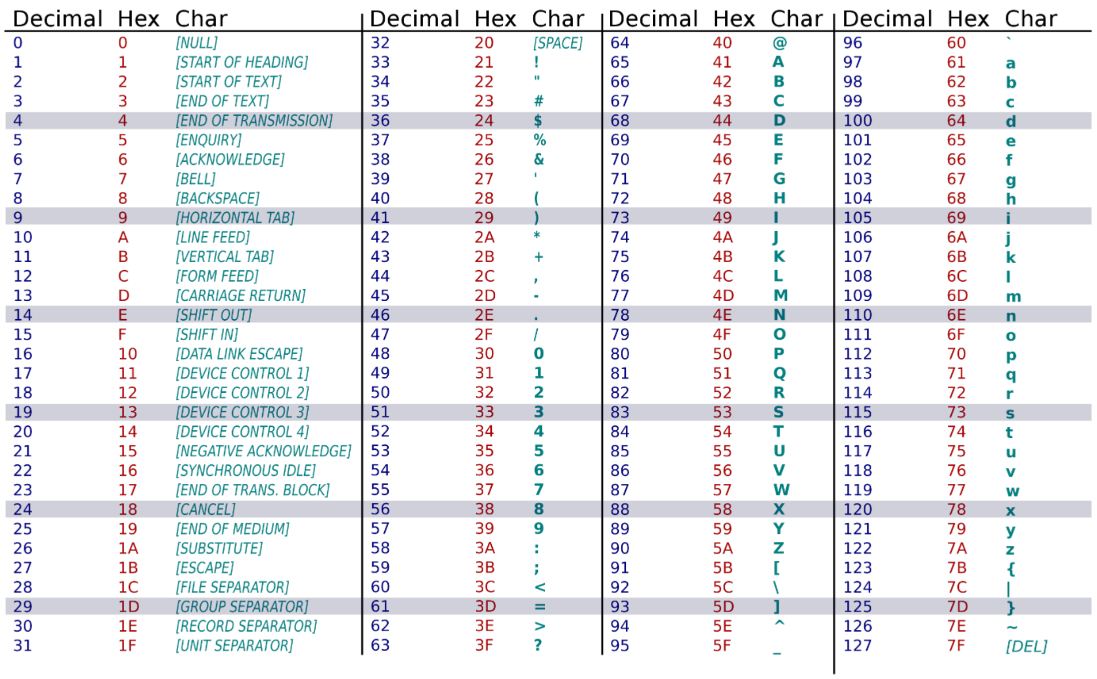
Table 5.1. Serial data packet.

Bytes	0	1	2	3	4	5	6	7
Component	STX	Command	ID	Data				ETX

The software interprets serial read and write data as ASCII encoded, as described in Table 5.3 [4]. For example, to represent the character "F" (value 15 or xF), the computer sends 70 (x46) to serial. One binary byte can contain one ASCII character. So, one byte (for the command and object ID) has a value range of 0-15 (x0-xF); four bytes has a range of 0-65535 (x0000-xFFFF).

[1] "x" denotes a hexadecimal number (base 16).

Table 5.2. ASCII table [4].



Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

There are 10 different commands, as shown in Table 5.3. There are 3 commands for testing the activity board, 6 for experiment setup, and 1 for performing the experiment. For the testing and setup commands, the software will send one packet to hardware, and then the microcontroller will send one packet back to send relevant data and to acknowledge that the initial packet was processed. The STREAM command is unique, when the hardware receives STREAM with True (1) in the data packet, it will write back at a known interval set by the SAMPLE RATE command. It will continue to write until the software writes STREAM with False (0) in the data packet. Currently, TEST LED, BATTERY, and CALIBRATE commands are unimplemented in hardware.

Table 5.3. List of commands and arguments

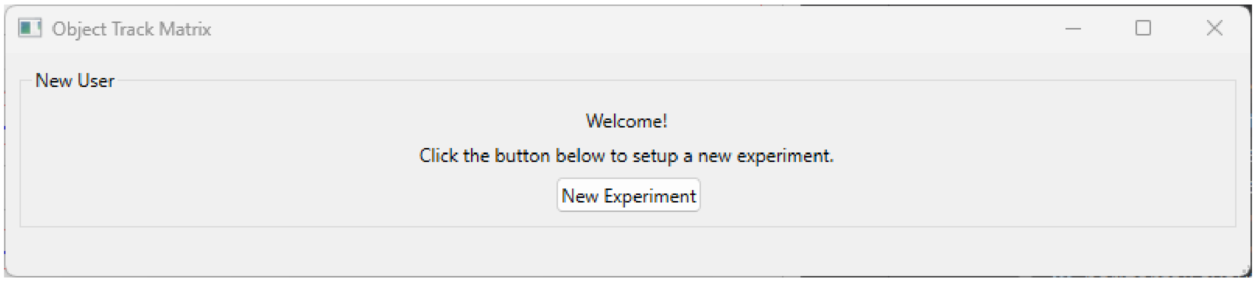
Name	Number	Software		Hardware		Description
Testing		ID	Data			
PING	0					call and response to test communication
TEST LED	1	Object ID	Time (ms)			Command that turns on all LEDs to test functionality
BATTERY	2	Object ID		Object ID	Battery percent	Get the percent of the battery life of one motion sensor
Setup						
CALIBRATE	3	Object ID				Sends the object ID to be calibrated. The respective motion sensor is moved, and its ID saved.
NTRIALS	4		Number of trials			number of trials in the experiment
TRIAL	5	Object ID	Trial number			Sets the object ID for the trial. several of these commands comprise of the experiment object sequence.
SEPARATION	6		Time (ms)			Sets the time between trials. This is the time

						between an LED off and on.
TIMEOUT	7		Time (ms)			Sets the maximum time an LED can be on for a trial
SAMPLE RATE	8		Frequency (Hz)			Sets the sample rate
Experiment						
STREAM	9		True / False	Object ID	Trial number	Start (true) or stop (false) the experiment. Sends the object ID (1-12) of the object that was moved and the trial number. Object ID is 0 if no object is moved

For commands that do not require an object ID or data, those portions of the packet will be zeros. This is not the most efficient use of bits, but it allows all packets to be the same length, which is easier to program and more human-readable.

5.2 Software User Interface

The software user interface (UI) is designed to be simple and intuitive. When first opening the software, the user is greeted by the welcome UI, as shown in Figure 5.2. Clicking the "New Experiment" button will change to the experiment details window (Figure 5.3).

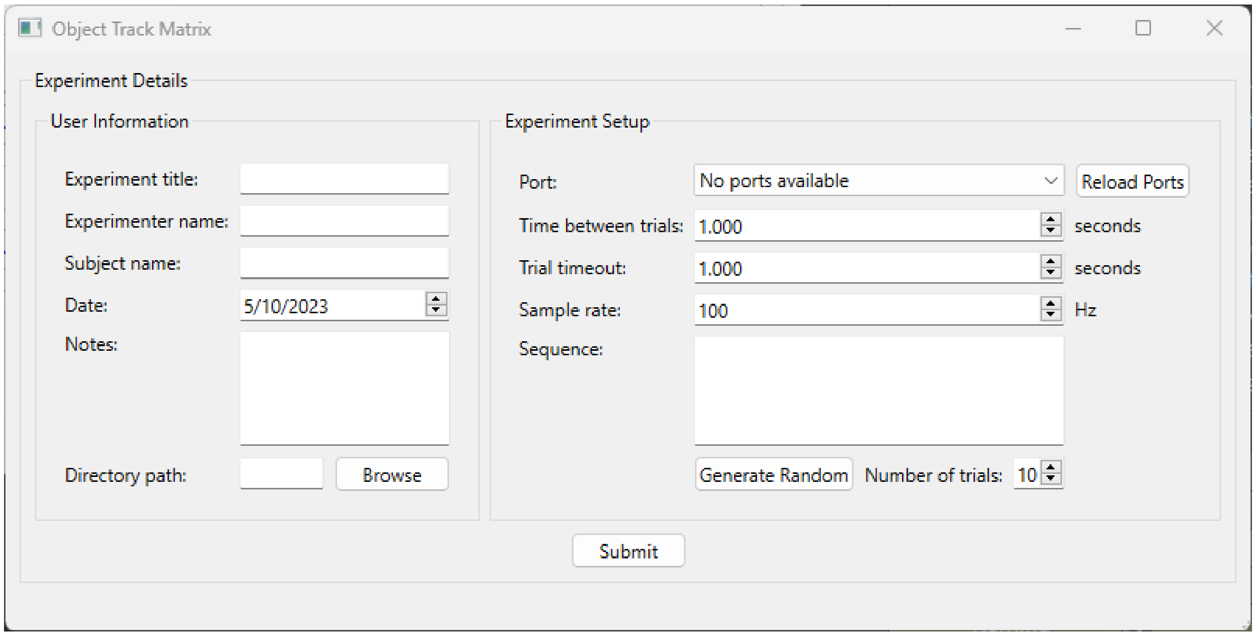


Welcome!

Click the button below to setup a new experiment.

Figure 5.2. Welcome UI [5].

In the experiment details UI (Figure 5.4), there are two group boxes containing various user input fields. The left box collects information about the user, including the title of the experiment, the name of the user/experimenter, and the name of the experiment's subject (who will interact with the activity board), the date of the experiment, any relevant notes, and the computer directory path to save data files to. Clicking the "Browse" button will open a window to the user to select a directory on their computer. The right group box collects information about the current experiment, including the serial port connected to the activity board, the time between trials, the time for a trial to end if there is no interaction (timeout), the data sample rate, and the sequence of trial objects. The user can write their own trial sequence or generate a random sequence by choosing the number of trials with the spin box and then clicking the "Generate Random" button. Clicking the "Reload Ports" button will check for the current serial ports available on the computer. Typically, ports names begin with "COM". The available ports on the user's computer can be found under the Device Manager on Windows. When the user has input all the required information, they can click the "Submit" button at the bottom of the window. If any required information was not given, "[!!!]" will appear on the right side of the invalid input field. If all information is correct, the experiment details window will lock. An "Edit" button will appear, which will unlock the experiment details window.



Experiment title:	
Experimenter name:	
Subject name:	
Date:	5/10/2023
Notes:	
Directory path:	

Port:	No ports available	
Time between trials:	1.000	seconds
Trial timeout:	1.000	seconds
Sample rate:	100	Hz
Sequence:		
Number of trials:	10	

Figure 5.3. Experiment Details UI [5].

After submitting the experiment details, the experiment group box will appear (Figure 5.4). The clicking the “Start Experiment” button will begin the experiment. First, the software will save the user information into the Experiment_Information.csv file. The software will write the experiment setup information to the hardware microcontroller and then begin streaming data. The microcontroller will send the current trial and object moved every t[s] = (sample rate [Hz])$^{-1}$; the software will save this data to the Experiment_Data_Stream.csv file. After all trials have finished, the software will analyze the experiment data to determine if each trial was successful (the correct object was moved) or failed and save this information in Experiment_Data_Stream.csv.

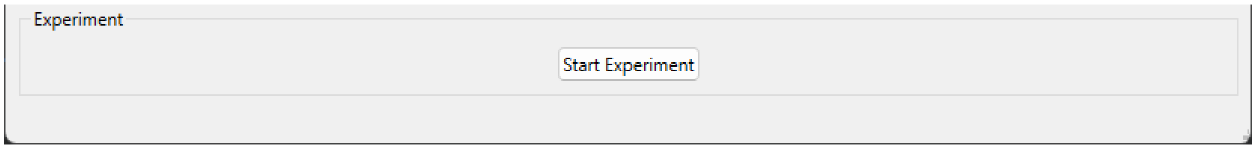


Figure 5.4. Experiment UI [5].

5.3 Software Internal Design

The software consists of six classes and three UI files, as described in Table 5.4. The main.cpp implementation creates an instance of the MainWindow class and shows the UI.

Table 5.4. Software class files.

Class Name	**Description**	**Files**
MainWindow	Top-level class and UI to request and submit experiment information, perform an experiment, and save data to files.	*.h/*.cpp/*.ui
GetUserInfo	UI with user input fields for the experiment title, experimenter name, subject name, date, notes, and directory path. This information is accessed using getter functions.	*.h/*.cpp/*.ui
ExperimentSetup	UI with user input fields for the serial port, time between trials, trial timeout, sample rate, and trial sequence. This information is accessed using getter functions.	*.h/*.cpp/*.ui
Commands	Handles building and interpreting serial packets. Also has functions to convert integers to hexadecimal strings, check the command parameter requirements, and get the number of bytes of the packet components.	*.h/*.cpp
SerialControl	Handles reading and writing command packets through a serial port.	*.h/*.cpp
ExperimentFileControl	Writes files to the experiment directory to store the experiment information, streaming data, and trial status as *.csv files.	*.h/*.cpp

The interconnections between the classes are shown in Figure 5.5. MainWindow is the toplevel class. GetUserInfo and ExperimentSetup are widget objects in the MainWindow UI. ExperimentFileControl and SerialControl are included in MainWindow, which has corresponding member variables. SerialControl includes the Commands class.

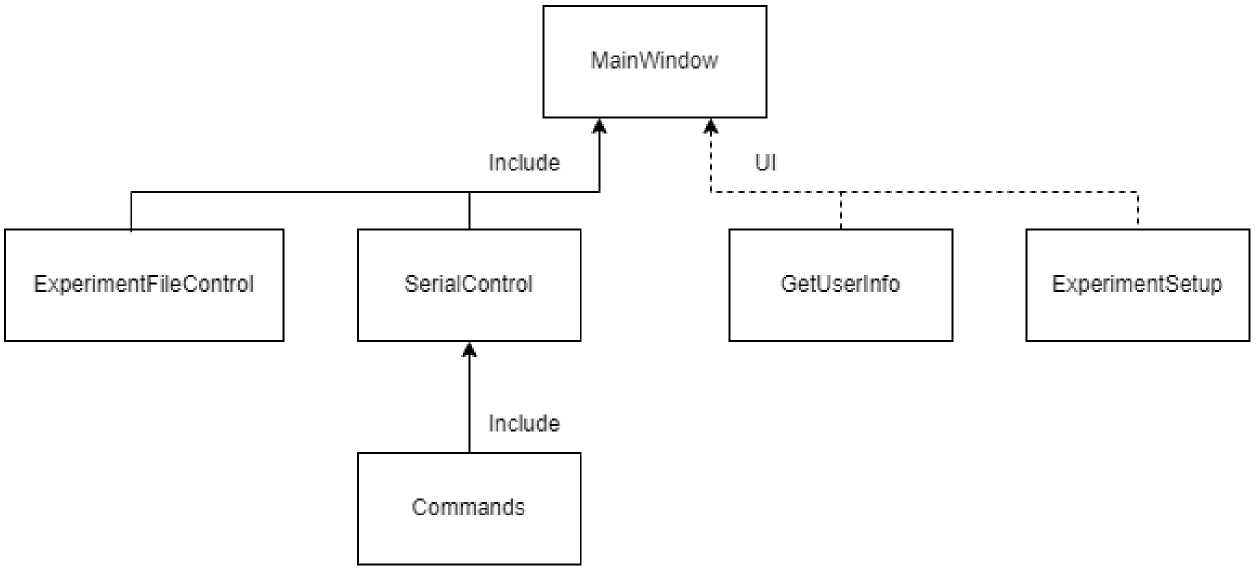


Figure 5.5. Class interconnections.

5.4 Microcontroller

The microcontroller interfaces between the software and the rest of the activity board hardware. I created functions to read binary packets, interpret them to usable values, perform the command, build binary packets, and write to software. The program has several variables to track the initialization conditions and experiment state (initialization or streaming). The microcontroller uses the arduino-timer library (Copyright (c) 2018, Michael Contreras) to send streaming data to the software at the set sample rate [7].

The main loop first checks if serial data is available. If so, it reads the binary command packet, and checks if the packet is valid. The program then performs that command. For initialization commands, the program updates the respective global variable. For the experiment commands, the program updates the device state. Next, the main loop checks if the state is in streaming mode. If so, it begins the experiment. For each trial, the trial's object LED ring turn on blue. If the correct object is moved, the LEDs flash green. If the wrong object was moved, the LEDs flash red. Throughout the experiment, the current trial and object moved is written to the software.

Code for the microcontroller can be found on both my and Ryan's GitHub [5,6]. Ryan has additional code for the Wi-Fi boards and motion sensors, which is discussed in his report.

6. Resources

The list of components for the object track matrix project is shown in Table 6.1. This includes the accelerometers, Wi-Fi boards, Arduino, cables, batteries, LEDs, 3D printing, and other assorted parts. The total cost for parts before tax is $333.37.

Table 6.1. Components and costs.

Item Name	Number of Units	Cost Per Unit	Cost	Link
MPU 6050	13	$3.33	$43.29	X
RGB LED (Pk of 100)	1	$8.99	$8.99	X
D1 Mini Micro Controller	13	$3.20	$41.60	X
D1 Mini LiPO Battery Charger	13	$2.99	$38.96	X
3.7V 500 mAh LiPo Battery	13	$7.25	$94.25	X
Micro USB Charging Cord	1	$9.36	$9.36	X
3D Printed Parts[2]	-	-	$20.00	-
Compression Spring[3]	24	-	-	-
Tapered Heat-Set Inserts M4 x 4.6 mm (pk 100)	1	19.67	19.67	X

[2] 3D printed parts were provided by Rob Young at the Instrumentation Design Laboratory. The amount listed reflects the physical materials used for printing.
[3] Provided from Ryan's personal stock.

Tapered Heat-Set Inserts M2.5 x 3.3 mm (pk 100)	1	14.11	14.11	X
M2.5x6 Screws (Pk of 100)	1	9.39	9.39	X
M4x50 Screws (Pk of 30)	1	9.99	9.99	X
AW9532	3	7.92	23.76	X
Various nuts, washers, heat shrink tubing & wire[4]	-	-	-	-
Total			$333.37	

7. Tests and Results

In the proposal, I suggested some tests to determine the success of the object track matrix. I have completed testing the timing delay and the experiment sequence

1. *Timing delay* (Section 7.1): the software will write a ping command to microcontroller. The microcontroller will then send back the same ping command to the software. The software will calculate the time difference from when the message was sent and received. The delay should not exceed 10 ms.
2. *Experiment sequence* (Section 7.2): the user will define an LED indication sequence then start the experiment. The user will then observe the activity board to ensure the same LEDs are activated and for the correct durations.

7.1 Timing Delay

To determine the time delay for reading and writing, I wrote a PING command to the software, read from hardware, and checked that the commands match. The program outputs the time delay in milliseconds to perform this operation. The computer has an error of 1 ms, as this is the smallest unit of measurement. I performed two tests, each with 20 trials. The results are

[4] Provided from Ryan's personal stock.

shown in Figure 7.1. The average time delay for all trials is 4±1 ms. This is within the 10 ms resolution requirement.

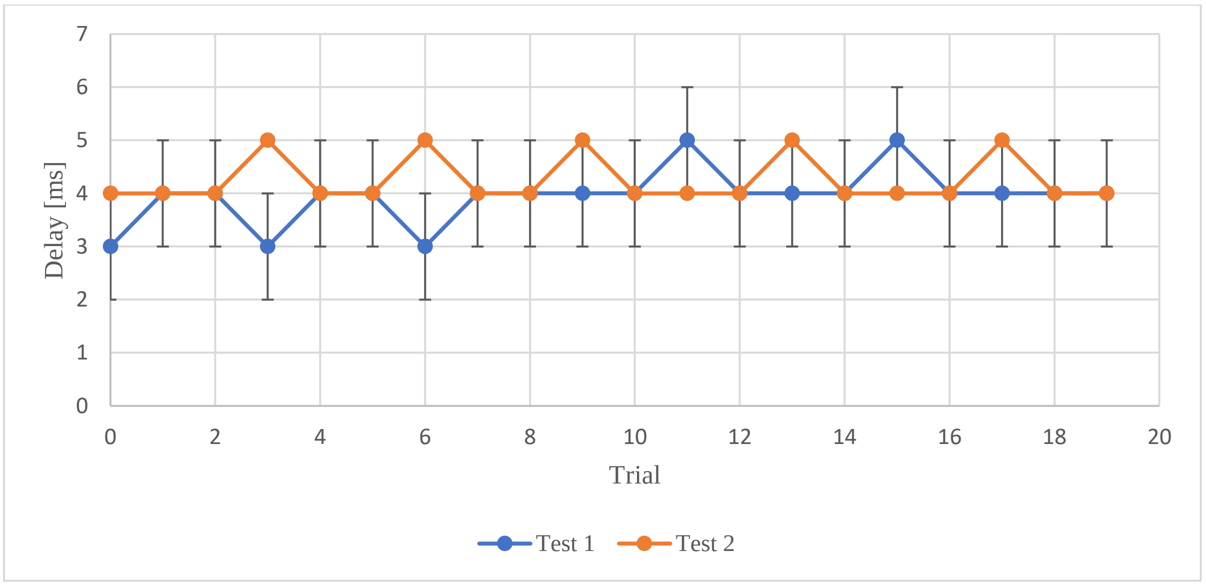


Figure 7.1. Timing delay for reading and writing.

7.3 Experiment Sequence

For this test, I submitted the following sequence in the experiment setup window: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. For each trial, the LED ring lights up. Then, I move the correct object. I track if the correct LED ring turns on and if the motion sensor operates correctly. These results are shown in Table 7.1. All objects operate correctly except for #12. This is due to a code bug when converting the integer 12 into a hexadecimal character "C"; there is a memory issue with the microcontroller, which causes the device to crash and reset. This forcibly ends the experiment. This issue is also discussed in Section 8.1.

Table 7.1. Testing each object.

Trial	Did the correct LED ring turn on?	Did the motion sensor activate when moved?
1	Yes	Yes
2	Yes	Yes
3	Yes	Yes
4	Yes	Yes
5	Yes	Yes
6	Yes	Yes
7	Yes	Yes
8	Yes	Yes
9	Yes	Yes
10	Yes	Yes
11	Yes	Yes
12	**No**	**No**

8. Future Work

There are several tasks for future work on this project, which include bug fixes (Section 8.1) and system upgrades (Section 8.2).

8.1 Bug fixes

One unusual bug is that the microcontroller resets when trying to convert an integer 12 to a hexadecimal character "C". To do this conversion, I use the sprintf() C++ function. This issues only occurs for 12, all other numbers have worked (as of now). Due to time constraints with this project, I eliminated object ID #12 from the software and microcontroller programming; only IDs #1 through #11 are valid.

Another issue is that the hardware will not write back the STREAM (data=false) command. This functionality is coded into the microcontroller program. The issue arose when merging my and Ryan's code. So, this bug is likely due to an unexpected control flow.

8.2 System Upgrades

The most scientifically important upgrade to the object track matrix system would be to ensure that there is no data loss. Currently, there are no checks to see if a streaming data packet was lost. There are two options to achieve this. One, refactor the serial communication protocol to include one byte to store the packet number. For each write back from the hardware, the packet number should increment by one and cycle from x0 to xF. Two, add a new command that sends every 100 STREAM packets. If the number of received packets is not equal to the number of written packets, then data has been lost. However, this second method may introduce additional delays.

Another upgrade to the object track matrix would be to implement the unused commands: TEST LED, BATTERY, and CALIBRATE. They are not required to run an experiment but would be useful for testing.

A useful feature to add to the system would be better handling of failed trials. Currently, if the wrong object is moved, the system will just move to the next trial. It may be useful to repeat the trial instead.

Another nice feature to add would be real-time data visualization. This could be as simple as displaying the current trial and object moved to the MainWindow UI. Or, the trial status could be plotted on a graph.

9. User Manual

Here are the steps to perform an experiment with the object track matrix:

1. Plug in the activity board USB to the user's computer.
2. The activity board will begin its setup procedure. Each LED ring will flash red, green, and blue one object at a time. Then it will test the motion sensors
3. Push each motion sensor as its LED ring turns on. Do this for all 12 objects.
4. Open the software on the user's computer.
5. Click the "Start New Experiment" button.

6. Complete the “User Information” and “Experiment Setup” fields. The sequence must contain only valid object IDs separated by commas.
7. Click the “Submit” button. If any of the required input fields is empty or wrong, the warning “[!!!]” will appear. Fix any issues and repeat this step.
8. If you wish to change the experiment details, click the “Edit” button. Go to step 6.
9. Click the “Start Experiment” button. This will begin the experiment on the activity board.
10. When an LED ring turns on blue, move the respective object. If the incorrect object was moved, the LEDs will flash red. If the correct object is moved, the LEDs will flash green.
11. Repeat step 10 for all trials.
12. At the end of the experiment, data will be saved into three files: Experiment_Information.csv, Experiment_Data_Stream.csv, and Trial_Status.csv will be saved to the “<Directory Path>/<Experiment Title>” folder.
13. Close the software.
14. Unplug the activity board USB from the computer.

Send any questions to ThresaKelly133pc@gmail.com or ryancoppens@ku.edu.

References

[1] KU Medical Center Department of Neurosurgery. (2023). Precision Neural Dynamics Lab. Retrieved February 1, 2023, from https://www.kumc.edu/school-of-medicine/academics/departments/neurosurgery/research /precision-neural-dynamics-lab.html

[2] Rouse, A. G., & Schieber, M. H. (2015). Spatiotemporal distribution of location and object effects in reach-to-grasp kinematics. Journal of Neurophysiology, 114 (6), 3268–3282. https://doi.org/10.1152/jn.00686.2015
https://journals.physiology.org/doi/full/10.1152/jn.00686.2015

[3] Qt (2023). Tools for each stage of software development lifecycle. Retrieved from https://www.qt.io/

[4] ZZT32, & Usha. (2023). ASCII Table. Retrieved from https://commons.wikimedia.org/wiki/File:ASCII-Table-wide.svg

[5] Kelly, T. (2023). Object Track Matrix. Retrieved from https://github.com/parallax-xallaraq/ObjectTrackMatrix

[6] Coppens, R. (2023). EPHSX601. Retrieved from https://github.com/xSYOTOSx/EPHSX601

[7] Contreras, M. (2018) arduino-timer - library for delaying function calls. Retrieved from https://github.com/contrem/arduino-timer