Instructions for getting started on your projects using the Arduino-Leonardo devices via MATLAB

1. Prerequisites

The following programs are required to run these experiments. It is recommended to install these in the same order as below.

- MATLAB
- 64-Bit GStreamer
- Psychtoolbox-3
- MATLAB Support Package for Arduino hardware
- Arduino IDE (Required for Heterochromatic Flicker Photometry only)
- The unzipped MATLAB code available on GitHub: *https://github.com/OxfordPerceptionLab/ICVS-Summer-School/Arduino-Leonardo*

This software (except for the MATLAB code) should all be installed on the ICVS summer school lab computers, but if you want to set up to run on your own computer you'll need to do the installs. You will need to download the MATLAB code from the GitHub yourself before you get started.

2. Introduction

As part of the ICVS summer school, each group of students will do a project based around an Arduino controlled LED-based stimulus device This document is a guide for using the Arduino Leonardo device to run colour vision studies, and it together with the associated software are intended to provide a point of departure for the projects. The broad options provided include a Rayleigh Matches, Heterochromatic Flicker Photometry, Brightness Matches, and Unique Yellow Settings. Each of these is implemented in a simple way with the provided software, and for each there are many possible extensions. Over the week, you will put together one of the devices and then have the opportunity to design/refine/run a pilot experiment with it. At the end of the summer school, each group will report on what they did and found.

This document describes Arduino device kits and how to assemble them, the provided experiments with pointers to the software, the setup required for each, and how to run the tasks through MATLAB, including the key binds for each task. The idea of this guide is just to get you started, there are many ways to refine and extend the experiments you can do with the devices.

Please be sure to read the document on human subjects ethics before collecting any data as part of your project. We are considering these projects to be teaching exercises, and for this reason the data may not be published. Ideas generated through your projects, however, are

fair game for follow up after the course - it's just the data collected here that may not be used or presented beyond the summer school itself.

This document is organized by starting first with a description of the experimental ideas and baseline implementations, followed by information on how to put together the Arduino devices from the kits we have provided, and then details on installing and using the software. As you do the projects, the summer school instructors will be available to answer questions, discuss ideas, troubleshoot, etc. There are additional documents related to calibration and colour space transformations that may be helpful as you proceed.

Finally, we note that this is the first time we have used these devices in a large-scale instructional context, and it is quite possible that you will need to troubleshoot hardware and software issues, make sense of documentation that may not be 100% consistent or complete, etc. Although this may produce some frustration, it is also the case that in doing research this is amost always the situation we find ourselves in, as part of discovery is figuring out how to make a set of equipment and ideas work for the first time.

This document as well as the software for the projects and other related documents are available on GitHub at:

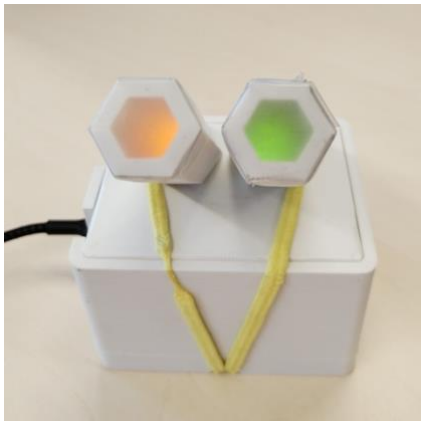
> https://github.com/OxfordPerceptionLab/ICVS-Summer-School/tree/main/Arduino-Leonardo

Please note that user files on the lab computers are wiped each night, so you will want to save your user files on a flash drive or other media at the end of each day.

3. The Arduino Leonardo device

The Arduino Leonardo is a portable device that can be used for colour vision-related experiments. It has two viewing hexagons – the left hexagon contains a yellow LED that acts as a reference light, while the right hexagon contains both red and green LEDs to be used as a test light. Some experiments, including Heterochromatic Flicker Photometry and Unique Yellow, only use the test light.

These LEDs are usually stable, but red and green LEDs that make up the testing light can also be set to flicker in counterphase, which is used for the Heterochromatic Flicker Photometry task (instructions to set this up are in section 4.1.1 below).



The MATLAB code provided allows the participant to change the intensity of these LEDs to complete different colour perception tasks. Although the code you have been sent will run the 4 experiments described, we highly encourage you to attempt to write some of your own code, or modify the provided code, to customise your use of the device. Some simple changes for beginners could be changing the key binds (see section 5 for the default key binds), using different starting values for the lights, or sequencing a series of experimental trials and saving the data.

Most alterations should be made in the main MATLAB file for each task - these are:

- RLM-Code/ArduinoRayleighMatch.m
- HFP-Code/ArduinoHeterochromaticFlickerPhotometry.m
- BRM-Code-ArduinoBrightnessMatch.m
- UQY-Code/ArduinUniqueYellow.m

respectively for each task. It is recommended that the other .m files in each folder are not altered in any way, unless you are experienced and comfortable coding similar experiments in MATLAB. The exception to this is changing the port number for the Heterochromatic Flicker Photometry task as required (see section 4.1.2).

4. Some Example Tasks

4.1. Task name abbreviations

In the GitHub folder and the MATLAB code provided, the task names are abbreviated as follows:

RLM = Rayleigh Match

HFP = Heterochromatic Flicker Photometry

BRM = Brightness Match

UQY = Unique Yellow

4.2. Task descriptions

4.2.1. Rayleigh Match description

This task uses both the test and reference lights. Participants are instructed to adjust the red/green balance of the test light, as well as the intensity of the reference light, until they look perceptually identical. Note that the red and green LEDs change together in this

experiment; increasing the intensity of the red LED simultaneously decreases the intensity of the green LED, and vice versa.

Thought question: What is being achieved by the Rayleigh Match? What do the results of the Rayleigh Match indicate about the participant? Is yoking the intensities of the red and green LEDs a good idea, and what would happen if you didn't?

4.2.2. Heterochromatic Flicker Photometry description

This task uses the test red and green LEDs only. The red and green LEDs turn on/off very quickly in counterphase, so that the light looks like it is flickering. Participants are instructed to adjust the intensity of the red LED until the light looks “stable” or “non-flickering”. The intensity of the green LED is constant for this task.

Thought question: What is being achieved by Heterochromatic Flicker Photometry? What do the results of Heterochromatic Flicker Photometry indicate about the participant? How would you expect the results to change if the red LED was held fixed and the green LED was adjusted, or if a different level of the fixed LED was used? How could you modify the experiment if you wanted to check whether flicker photometric matches are linear?

4.2.3. Brightness Match description

This task uses both the test and reference lights. The intensity of the yellow reference light is set by the researcher. Participants are instructed to adjust the intensity of the test light until the two lights look “equivalently bright”. Participants first complete this task with the red test LED only, then the green test LED only, per trial.

Thought question: What is being achieved by the Brightness Match? What do the results of the Brightness Match indicate about the participant? Would you expect the results to be predicted in any way by the flicker photometric experiment? What would happen if you matched brightness to mixtures of red and green, rather than red and green alone?

4.2.4. Unique Yellow description

This task uses the red and green LEDs only. Participants are instructed to adjust the intensity of the red test LED and the green test LED separately until they perceive the light to be a “unique yellow”, or a “pure yellow”. This is a yellow that does not perceptually appear to have any red or green light mixed in. Note that the intensity of the red LED and green LED change independently in this experiment; changing the brightness of the red LED does not affect the brightness of the green LED, and vice versa.

Thought question: What is being achieved by the Unique Yellow task? What do the results of the Unique Yellow task indicate about the participant? What would happen if you fixed the intensity of one of the LEDs and adjusted the other?

4.3 Discussion

The four experiments above represent simple versions of their respective measurement questions, and in each case we have provided some discussion questions to help you think about extensions. These too are just examples, and for your project you may want to go beyond the ideas presented here.

Thought questions: Do you have any ideas about other tasks that could be run using the current device setup? Do you have any ideas about small additions/changes the hardware/software that would allow the device to perform additional tasks?

In addition, the example programs specify the stimuli directly in terms of the DAC values passed to set the current to each LED. As discussed in the light measurement and calibration handouts, to understand the results in terms of visual processing, you would want to measure and correct for the gamma functions of the LEDs, and know the spectra emitted by each. We encourage you to do this. This would allow you to, for example check the degree to which your measurements of some aspect of perception match up with what is expected from the literature.

5. Putting Together the Arduino

The Arduino devices are provided as kits, and the first step is to assemble them. We recommend that you watch the video available here, which shows one of them being put together.

> https://onedrive.live.com/?authkey=%21AOk2AyiAh6hIyBw&cid=89B07BAA70C581C2&id=89B07BAA70C581C2%2142266&parId=root&o=OneUp

One important step is shown only symbolically in the video - that is cutting pieces of wratten filter to fit into the junction between the two pieces of the viewing tube. This takes a little practice. One way to do it is to use the view tubes to make an impression in the filter, and then cut on the outline. Thought question. *What might be the purpose of inserting the orange filter into the viewing tube?*

Another thing to think about is how to make the light coming out of the viewing tubes spatially uniform. We have found that scotch tape can work for this purpose, but there are probably other approaches.

You will not need to build the Arduino device from scratch for this course, but if you would like your own afterwards, instructions and parts lists are available here:

https://github.com/BrainardLab/TeachingCode/tree/master/ArduinoAnomaloscope/xxxMakingOne

Once you've assembled the kit, plug it into the USB port of the ICVS lab computer. You should then be able to run any of the experimental scripts described below and observe the lights in the device behaving as expected.

6. Software Setup instructions

To run the tasks from your computer, plug in the Arduino Leonardo device using either a USB-A or USB-C connector before you start. The Arduino device uses a micro-USB connection and can be connected to a computer as shown in the attached photograph attached.

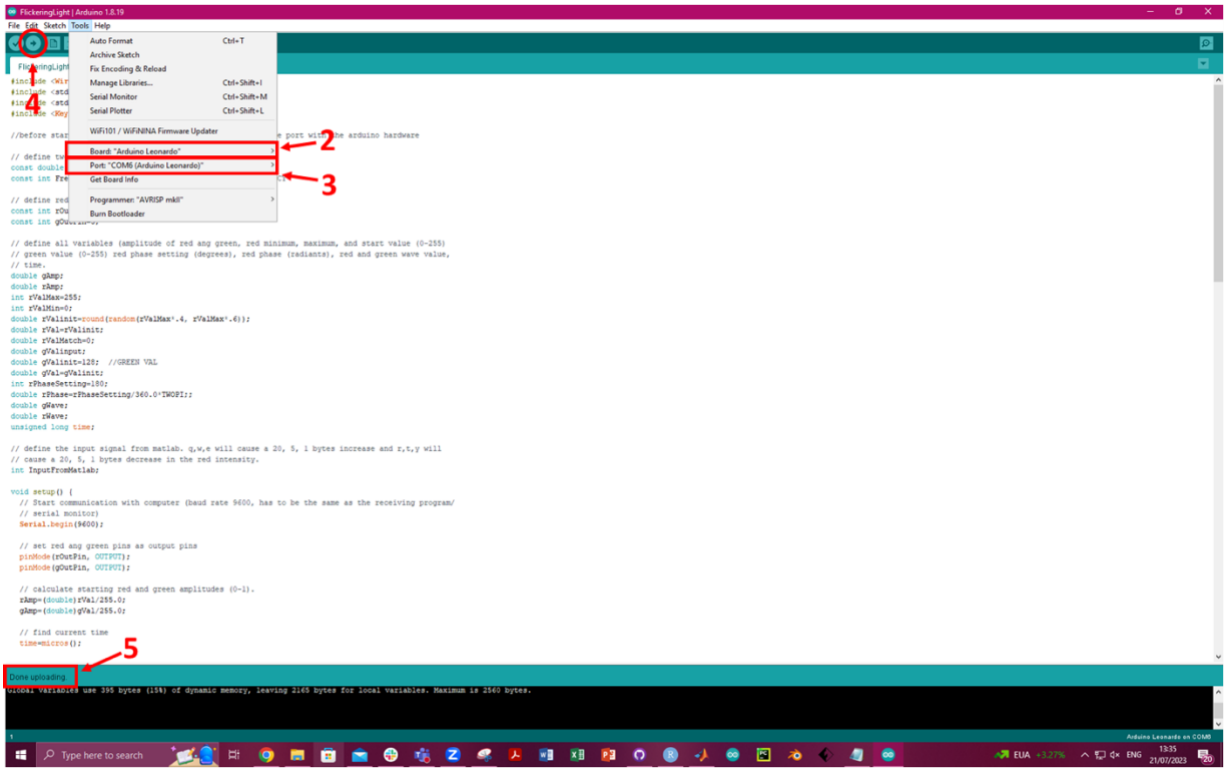
6.1. Heterochromatic Flicker Photometry setup instructions

If you are planning to run the Heterochromatic Flicker Photometry task for your experiment, you will need to follow the additional steps in section 4.1 before moving on to the general setup steps in 4.2. For all other tasks, please skip to section 4.2. Indeed, we recommend that you first skip the rest of 4.1 just to make sure your device is working as expected before coming back to 4.1.

6.1.1. Programming the test lights to flicker in counterphase

MATLAB does not provide sufficiently fast temporal control of the Arduino in order to make the lights flicker at high enough frequencies for a HFP task. *Thought question: what frequencies might be best suited for the HFP task and why?* In order to achieve the necessary temporal control one must therefore drive the Arduino via the Arduino IDE to create the counterphase flicker and use MATLAB to interface with the Arduino IDE to control the intensity of the lights. To set-up the control via Arduino and interface via MATLAB you need to follow the steps below:

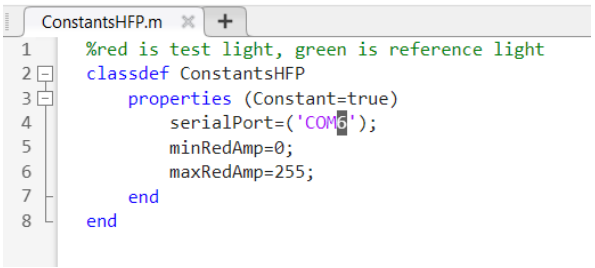
1. Open FlickeringLight.ino from the GitHub repository in Arduino IDE. This file can be found in Arduino-Leonardo/HFP-Code/FlickeringLight/FlickeringLight.ino in the downloadable GitHub folder.
2. In the “Tools” tab on the top bar, select “Board”, then select option “Arduino Leonardo”.
3. In the “Tools” tab, select “Port”, followed by the port name that ends with “(Arduino Leonardo)”. In the example below, this is port “COM6”.
4. Click the right-pointing arrow on the blue bar to upload this code to the device.
5. Once the bottom blue bar reads “Done Uploading”, you can close or minimise Arduino IDE.



6.1.2. Directing MATLAB to the correct port number

Once this code has finished uploading to the device, you will also need to check that MATLAB is opening the correct port to use the Arduino.

1. Check which port the Arduino device is using (instructions for Windows, Mac, and Linux are here: *https://uk.mathworks.com/help/supportpkg/arduinoio/ug/find-arduino-port-on-windows-mac-and-linux.html)*
2. Open ConstantsHFP.m in MATLAB. This file can be found in Arduino-Leonardo/HFP-Code/ConstantsHFP.m in the downloadable GitHub folder.
3. Change the name in the serialPort variable to the port number the Arduino is using (e.g. `'COM6'` or `'COM20'`). Make sure to only change the number after "COM" and nothing else in this file!
4. Save changes to this file – you can also close this file once the changes have saved.



6.2. Setup instructions for all tasks

1. Open MATLAB 2022a.
2. Set working directory to the downloaded GitHub folder, called “Arduino-Leonardo”; if you’re using GitHub Desktop on Windows this will be in “C:/Users/your_username_here/Documents/GitHub/ICVS-Summer-School/Arduino-Leonardo/”).
3. From the command window, run the task using the command “RunColourVisionStudy”. The first input variable should be the task abbreviation inside apostrophes, and the second input should be the number of trials. Some examples are below.

```
>> RunColourVisionStudy('RLM', 1)    % Runs 1 Rayleigh Match trial

>> RunColourVisionStudy('HFP', 3)    % Runs 3 Heterochromatic Flicker
Photometry trials

>> RunColourVisionStudy('BRM', 8)    % Runs 8 Brightness Match Trials

>> RunColourVisionStudy('UQY', 10)   % Runs 10 Unique Yellow trials
```

4. Once the study is running, MATLAB will ask you to input a participant ID. Enter any unique sequence of numbers and/or letters to identify that participant as a string, e.g. ‘ICVS1’, then press “return” on the keyboard.
5. If you are running the Brightness Match experiment, MATLAB will also ask you to input a brightness value for the yellow reference light. This should be an 8-bit value specified as an integer between 0 and 255. Enter the value you’d like to use, then press ‘return’.
6. The first trial will then start for the participants to complete. Once the study is complete, or if it is exited early using the correct key bind, the results with be saved in the “Saved-Data" folder, in the subfolder corresponding to your chosen task. The participant ID will be in the file name. Note that these computers delete data once shut down so be sure to take a copy of your Saved Data folder on a USB stick or hard drive.

6.3. Switching between tasks

Note if you switch between whether MATLAB drives the Arduino (as in the RLM, BRM, UQY tasks) and Arduino IDE driving the Arduino (as in the HFP task), MATLAB needs to set-up a new connection with the Arduino Leonardo board. It does so automatically when you switch from RLM/BRM/UQY -> HFP (after you have initialised control via the Arduino IDE as described in Section 6.1.1.), but when switching from HFP -> RLM/BRM/UQY it needs to run the program twice to re-establish the new connection. So when you try to run the RLM/BRM/UQY study after the HFP study, and you will find an error message occurs after a few minutes. Simply re-run the code to start the task in the command window and it should then work fine with no error message on the second attempt.

7. Key binds for running the experiments

Note: the “test light” refers to the red and green LEDs in the right viewing hexagon, and the “reference light” refers to the yellow LED in the left viewing hexagon.

7.1. Rayleigh Match key binds

7.1.1. Participant key binds for Rayleigh Match

‘A’ = increases the red/green balance of the test light

‘D’ = decreases the red/green balance of the test light

‘W’ = increases the brightness of the reference light

‘S’ = decreases the brightness of the reference light

7.1.2. Experimenter key binds for Rayleigh Match

‘K’ = decreases the step size for changes in the test light

‘L’ = decreases the step size for changes in the reference light

‘O’ = check the current values of the lights without finishing the trial

‘RETURN’ = finish the trial

‘I’ = resets the trial without saving

‘=+’ = safely exits the experiment, saving all completed trials

7.2. Heterochromatic Flicker Photometry key binds

7.2.1. Participant key binds for Heterochromatic Flicker Photometry

‘A’ = increases the brightness of the red test light

‘D’ = decreases the brightness of the red test light

7.2.2. Experimenter key binds for Heterochromatic Flicker Photometry

‘K’ = decreases the step size for changes in the red test light

‘O’ = check the current values of the lights without finishing the trial

‘RETURN’ = finish the trial

‘I’ = resets the trial without saving

‘=+’ = safely exits the experiment, saving all completed trials

7.3. Brightness Match key binds

7.3.1. Participant key binds for Brightness Match

‘W’ = increases the brightness of the current test light

‘S’ = decreases the brightness of the current test light

7.3.2. Experimenter key binds for Brightness Match

‘K’ = decreases the step size for changes in the current test light

‘O’ = check the current values of the lights without finishing the trial

‘RETURN’ = finish the trial

‘I’ = resets the trial without saving

‘=+’ = safely exits the experiment, saving all completed trials

7.4. Unique Yellow key binds

7.4.1. Participant key binds for Unique Yellow

‘A’ = increases the brightness of the red test light

‘D’ = decreases the brightness of the red test light

‘W’ = increases the brightness of the green test light

‘S’ = decreases the brightness of the green test light

7.4.2. Experimenter key binds for Unique Yellow

‘K’ = decreases the step size for changes in the test light

‘O’ = check the current values of the lights without finishing the trial

‘RETURN’ = finish the trial

‘I’ = resets the trial without saving

‘=+’ = safely exits the experiment, saving all completed trials

Arduino Projects Guide