Written by Raquel Ibáñez Alcalá
Device: Texas Instruments MSP430-FR2355 development board running RECORD Firmware
Created 8/May/2023, last updated 1/Jun/2023 (Rev. 1.1)

R.E.C.O.R.D.

(Reward-Cost in Rodent Decision-making)

User Manual

Revision 1.1

Table of Contents

Background information

This guide will talk specifically about the Texas Instruments MSP-EXP430FR2355 launchpad development kit (https://www.ti.com/tool/MSP-EXP430FR2355) and about microcontroller features specific to the RECORD system. Pin numbers, peripherals, and connections will differ between microcontrollers from the same and different manufacturers, and we cannot guarantee that this guide will be accurate for any other microcontroller. For a detailed user guide on this microcontroller, please see the manufacturer user guide ((https://www.ti.com/product/MSP430FR2355).

How LED brightness is controlled

The RECORD system microcontroller uses Pulse Width Modulation (PWM) to regulate the brightness of the LEDs which encode cost levels. A total of four PWM signals are produced by the microcontroller, one for each feeder. The amount of time within the PWM signal's period that the signal is on (logical high) is defined as the **duty cycle**. A higher duty cycle will result in a brighter LED, while a lower duty cycle will result in a dimmer LED. The **duty cycle** is dependent on two **Capture/Compare Register (CCR) values**; CCR0 and CCR1, which are defined in the microcontroller firmware.

The duty cycle is defined by the formula:

$$DC(\%) \; = \; 1 - \frac{on\ time}{signal\ period},$$

where *signal period* is the length of time it takes for the signal to complete a full cycle, and *on time* is the amount of time the signal is held at a logical high. We can expand this formula to be in terms of our two **CCR values**:

$$DC(\%) \; = \; 1 - \frac{CCR1}{CCR0},$$

Where CCR0 defines how many **ticks** it will take to count up to the signal's period and sets the signal to a logical high, and CCR1 is how many **ticks** it takes to bring the signal back down to logical 0.

To explain **ticks**, we must first talk about the microcontroller's internal clock. The microcontroller chip runs at a certain speed determined by an internal clock, which is driven by a crystal oscillator. In the RECORD system, the oscillator signal is internally divided and fed to the system clock, which produces an 8 MHz clock signal, this is set as the **system clock**. Each cycle in the system clock signal is defined as a **tick**.

With the system clock set to a frequency of 8MHz, we can define the period of the PWM signal as 1 millisecond by setting CCR0 to a value of 8000. We determine this value by using the formula:

$$CCR0 \;=\; \frac{Desired\ PWM\ period}{Clock\ period}$$

We want a PWM period of 1 millisecond since this value is relatively easy to work with, so plugging the values in , we get:

$$CCR0 \;=\; \frac{0.001}{8,000,000^{-1}} \;=\; 8000.$$

CCR1 is a similar but user-variable value which defines the amount of **ticks** it should take for the signal to be set to a logical low. This in turn defines the width of the signal pulse. As CCR1 approaches CCR0, the width of the pulse will increase, making the LED glow brighter, as CCR1 approaches 0 the shorter the pulse will be and the dimmer the LED will glow. Since CCR1 is the only parameter that can be adjusted, we will refer to it only as the **CCR value**. Thus, the driving factor in regulating the cost intensity is the **CCR value**, a whole number ranging from 0 to a maximum of 8000.

In summary, the brightness of the cost LED can be controlled by varying how long in the signal period the signal is kept at logical high vs logical low. This is defined as the duty cycle of the PWM signal. The rest of this guide will be referencing the **CCR value** for instructions.

Lastly, we must relate LED brightness (Lux) to the **CCR value**. It is also important to relate these two parameters to the duty cycle of the PWM signal, as the brightness of the cost LED will be dependent on the shape of the PWM signal and not voltage. We do this in our cost LED calibration document, linked below:

LED Calibration Data - Google Sheets

What are cost levels?

Cost levels are discrete brightness levels that the microcontroller can set. As mentioned before, cost levels are dictated by the **CCR value**. It is important to note that the **CCR value** and **LED brightness** have an inverse relationship, meaning that as the **CCR value** approaches its maximum of 8000, the **LED brightness** will approach 0, or the LED will be dimmer until it turns off. Cost levels are measured in **Lux**, which is a distance-dependent brightness unit measured using a **lightmeter**. We typically measure Lux at 3 mm or less away from the LEDs. Each cost level is calibrated to a range of Lux values. In rodents, bright light induces an aversive reaction, which is why light was chosen to serve as a cost in our cost/benefit tasks. **Brightness requirements may change from rat to rat, for this reason, it is important to calibrate!**

The following are general guidelines of how many Lux are required for each cost level. Always check with your section supervisor or the PI before determining what is needed at each level.

- Cost level 0 **(L0, 0 Lux)** represents an LED which is off. This presents no cost to the rat in the arena. It is typically not used in an experiment setting outside of simply turning lights off, though some special cases may apply. This level is not configurable.

3

- Cost level 1 **(L1, 7-15 Lux)** corresponds to the lowest brightness . This presents a low cost for the rat in the arena to obtain its reward. We typically keep this level low enough to not scare a rat, but high enough for them to see the light.

- Cost level 2 **(L2, 40-60 Lux)** corresponds to an intermediate brightness between the lowest and highest. It is associated with a mid-level cost. This level should be high enough to be somewhat uncomfortable, but not so bright that the rat never approaches offers associated with this cost.

- Finally, cost level 3 **(L3, 140-? Lux)** corresponds to the highest (safe) brightness for our experiments. The highest Lux that an LED can output will depend on the LED itself, but it should never shine so bright that the rat is traumatised by it and becomes conditioned to hate running our experiments.

Cost levels are calibrated using the program PuTTY, which has a command line analogue called Plink. This guide has a section on using PuTTY to explain how calibration and reconfiguration is done, as its graphical user interface is somewhat friendlier for the average user.

What are reward levels?

Reward levels are not so much dictated by the microcontroller, but rather the rewards made available on the RECORD arenas. However, the microcontroller sees reward levels as the relay which activates the solenoid valve that dispenses a reward to a particular feeder, thus it is extremely important that each solenoid valve and feeder is associated with the correct relay port. When it comes to reward, the only thing the microcontroller can do is select a particular feeder to dispense reward to, and hold a relay closed for a set amount of time so that the solenoid valve dispenses more or less solution into a feeder (see "Configuring relay active time").

Communicating with the RECORD microcontrollers

## Using PuTTY to communicate with RECORD

PuTTY is an SSH/Telnet client which also has the capability to serve as a serial communications interface between a PC and a serial device connected through a USB COM port, such as a microcontroller running the Universal Asynchronous Receive Transmit (UART) protocol. *In other words, you use PuTTY to communicate to the RECORD system microcontrollers while they are connected via USB cable to a PC.* Multiple PuTTY sessions may be loaded at the same time to communicate with each microcontroller, but two sessions cannot communicate with the same microcontroller. *This section will walk you through how to go through the process of calibrating each feeder to a desired lux level.*

This guide assumes a Windows PC with Code Composer Studio (https://www.ti.com/tool/CCSTUDIO) and PuTTY (https://www.putty.org/) already installed is being used, and that a microcontroller running *firmware version v1.2 or later* has been plugged into your PC.

Setting PuTTY up for the first time

1. On the RECORD microcontroller, make sure that the top RXD and TXD pins on the top board-to-board header are jumped to P1.6 and P1.7, respectively.
2. Connect the RECORD microcontroller to your computer via USB.
3. On Windows, open the device manager by pressing and holding the Window key and pressing the 'R' ('Run') key to open the run dialog. Type 'devmgmt.msc' into the dialog box and press Enter.
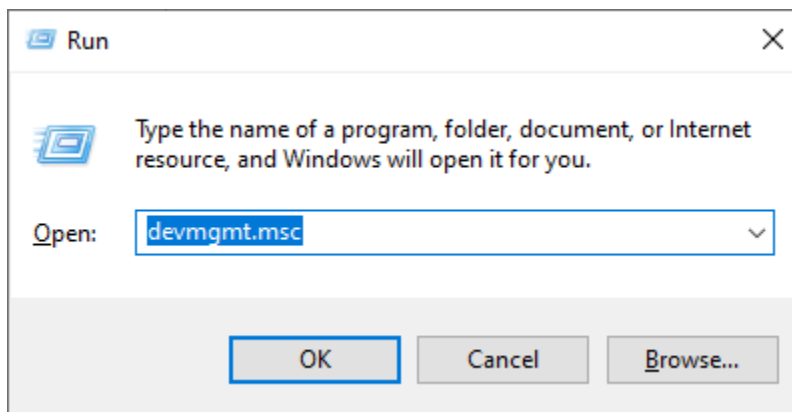
4. Navigate down to 'Ports (COM & LPT)' and expand the dropdown menu. Look for 'MSP Application UART1' and take note of the COM port number listed here (in parenthesis). If you do not see this entry, make sure that you have Code Composer Studio with the MSP430 library and drivers installed in your system (https://www.ti.com/tool/CCSTUDIO).

> Monitors
> Network adapters
∨ Ports (COM & LPT)
    Communications Port (COM1)
    MSP Application UART1 (COM11)
    MSP Debug Interface (COM10)
> Print queues
> Processors
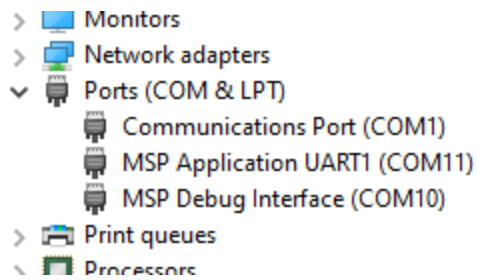
5. Open PuTTY and under 'Connection type', select 'Serial'. This is under the 'Session' menu.
6. Change 'Serial line' to the COM port that you took note of in step 4.
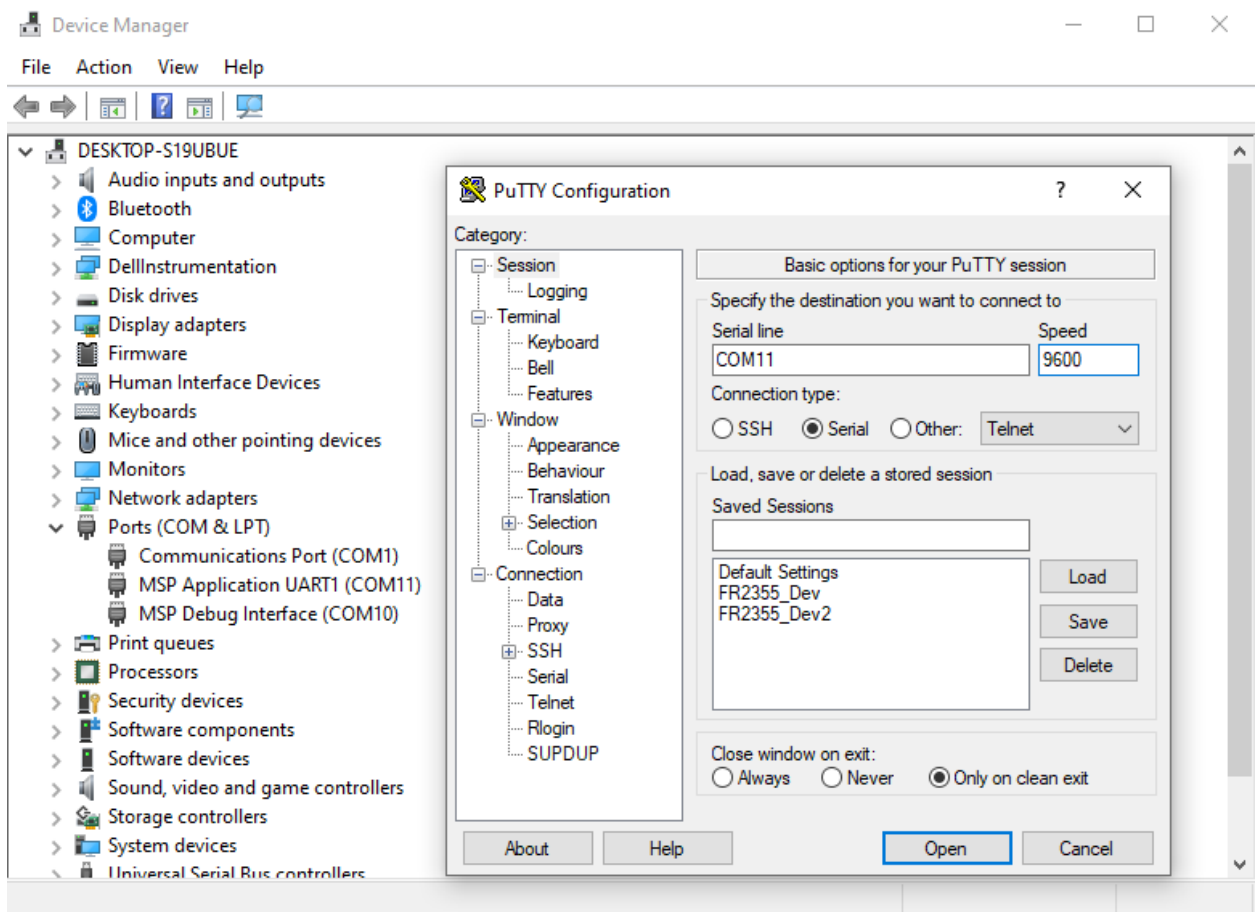
7. Navigate to 'Connection > Serial' and change 'Flow Control' to 'None'.
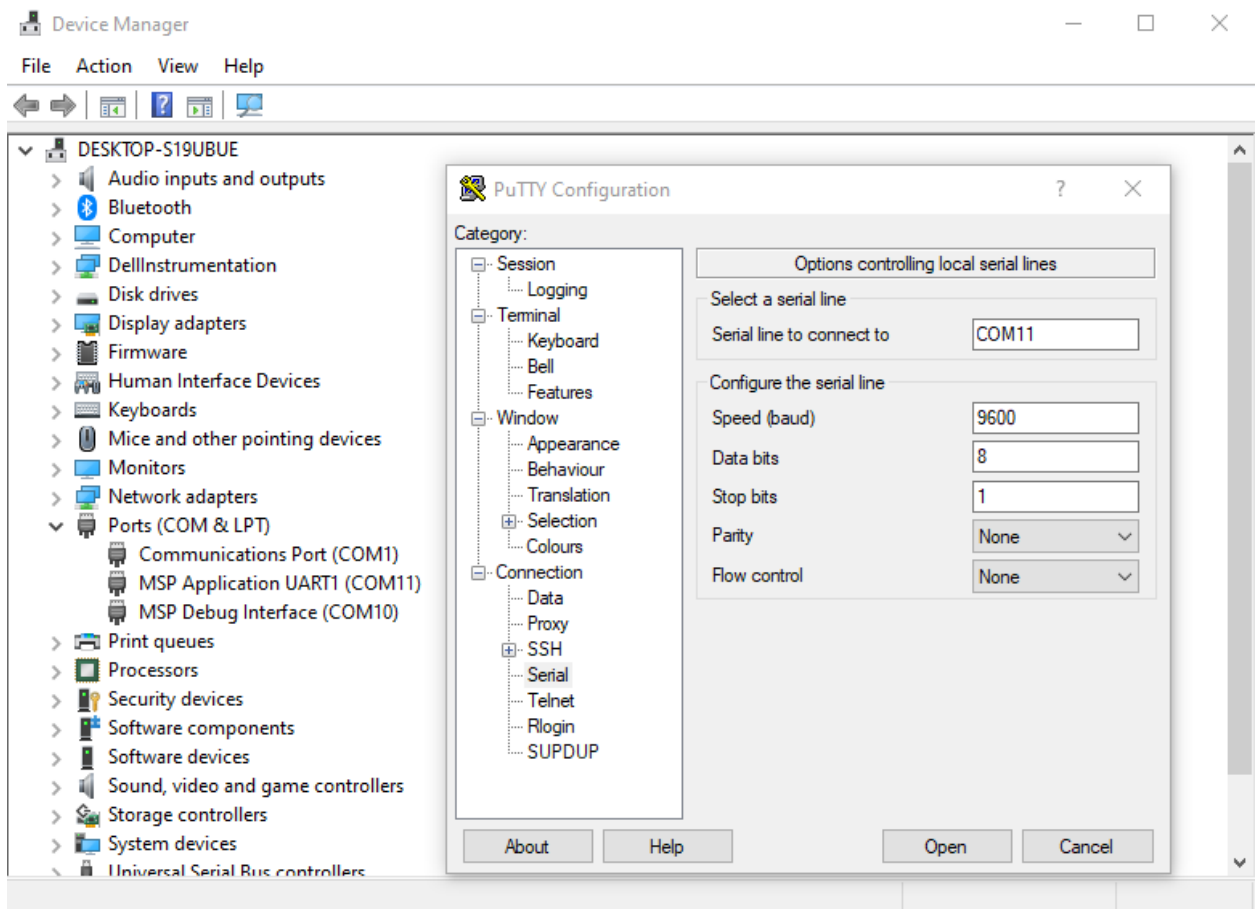
8. Finally, navigate back to the 'Session' menu and under 'Load, save or delete a stored session > Saved Sessions', give this preset settings profile a name then click 'Save'. You will be referencing this profile name often.
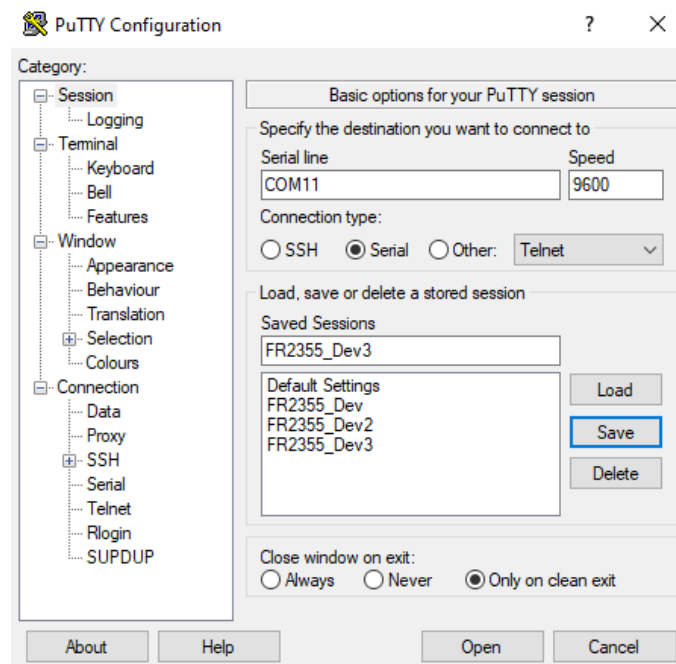
9. Click 'Open' and a black command window should appear. Try sending a capital 'R' ('Reset') to see if the device responds.

Opening PuTTY and connecting to an arena microcontroller

1. Turn on the microcontrollers. Do so by connecting them to the PC via USB. In our case at the Friedman Lab, do so by pressing the rightmost button on the USB hub located to the left of the computer tower in the recording room. The USB hub will light up with blue light indicating that it is on.

2. Open PuTTY. You will likely find this program in a folder named "PuTTY (64-bit)" under the Windows Start menu. Otherwise, you may search for it by typing "putty" in the Windows search bar.

3. Once open, PuTTY will load a default session preset. Near the bottom you will find a white space containing other session presets, and at the very bottom of the windows, you'll find an "Open" button. DO NOT click the "Open" button yet!
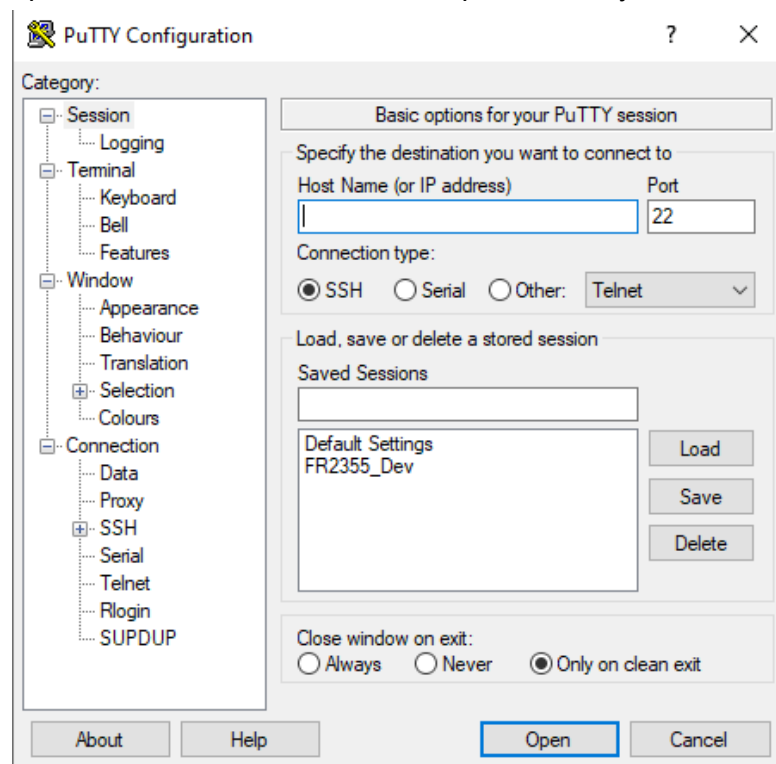
4. Each preset will configure PuTTY to work with its corresponding arena. Double click the preset that corresponds to the arena that you are calibrating, this will open a communication channel to that arena's microcontroller.

   a. FR2355 - Arena 1
   b. FR2355_2 - Arena 2
   c. FR2355_3 - Arena 3
   d. FR2355_4 - Arena 4

5. A small black window with a green cursor and no text will appear. This is where commands are sent to the microcontroller. With the window in focus, type the letter 'R' (capitalised). The microcontroller will respond by repeating what you sent, along with a message, which will change depending on what you send. 'R' is a command that resets the microcontroller to its idle state, turning all LEDs off. The black PuTTY window should now display the message "all off"; if no message is displayed, make sure the microcontroller is powered and connected to the PC, then repeat steps 2 through 4 and try again. **Contact a lab supervisor or the PI if there is still no response, DO NOT tamper with the microcontroller on your own.**

6. You've successfully connected and are now communicating with the microcontroller! Continue to the next section to calibrate or configure LED brightness.

Using Plink to communicate with RECORD

RECORD can be controlled via command prompt if Plink, PuTTY's command line executable, is installed. Once a PuTTY session is created following the steps in the previous two sections, one may opt to interfacing with the system by opening a command line window and typing

<div align="center">

`plink -load [session name]`,

</div>

After which the command line will behave just the same as any PuTTY command line.

Using RECORD-lib to communicate with RECORD

RECORD-lib is a Python (https://www.python.org/) library custom made for communications with the RECORD system. It uses the 'pyserial' library as a basis and includes various functions to do things like set a cost light up, deliver a reward, etc. This along with the 'Trials' class was used to create a Python script that can be run through a command line interface or through an IDE like Spyder (https://www.spyder-ide.org/) which runs through our high- and low-cost cost/benefit tasks, as well as our association task through tweaking of a few parameters. You can find RECORD-lib in our Github repository (https://github.com/rjibanezalcala/RECORD/tree/main/python/RECORD-lib) along with our sample trial (https://github.com/rjibanezalcala/RECORD/blob/main/python/RECORD-lib/sample_trial_with_output_v0.3.5.py). Here we'll describe what each parameter is.

Currently, the following trial parameters must be modified in the code itself, so a graphical IDE is recommended when running trials. After all parameters have been set, click the 'Run' button or press 'F5' to run the script.
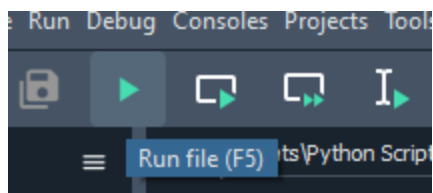
Make sure to set the 'mcu_com_port' parameter to the COM port the UART interface of the microcontroller is in. See steps 1 through 4 under 'Setting PuTTY up for the first time' for more details.

```
97      mcu_com_port = "COM4"
98      TTL_ON        = False
99      # Parse incoming trial p
```

```python
57      # BE SURE TO NOT REMOVE QUOTATION MARKS WHEREVER YOU MAKE CHANGES!!
58      trialParams = {
59                  # Time intervals, in seconds.
60                  'inter_trial_interval': 1,
61                  'decision_interval'   : 1,
62                  'feeding_interval'    : 1,
63                  # Number of trials
64                  'trials'              : 5,
65                  # Do not need to be changed
66                  'levels'              : [0, 1, 2, 3],
67                  'feeders'             : ['a', 'b', 'c', 'd'],
68                  # Probability of a COST level appearing in the trial list. Sum of
69                  'lvl_probs'           : [0, 1, 0, 0],
70                  # Probability of a FEEDER appearing in the trial list. Sum of all
71                  'fdr_probs'           : [0.25, 0.25, 0.25, 0.25],
72                  # Sugar concentration in reward solution. If compound solution (fo
73                  'rewards'             : ["9%","5%","2%","0.5%"],
74                  # Reward volume delivered
75                  'reward_volume'       : "4mL",
76                  # Cost intensity, in lux. Include units, no spaces.
77                  'intensities'         : ["0lux","15lux","140lux","290lux"],
78                  # Subject's name, as defined in rat roster.
79                  'subj_name'           : "A4",
80                  # Specify the type of manipulation done. If regular behaviour tria
81                  'subj_health'         : "Normal",
82                  # Specify the subject weight, in grams. Include units, no spaces.
83                  'subj_weight'         : "Undefined",
84                  # Specify the type of task done. For example L1, L2, L3, L1L2, L1L
85                  'task_type'           : "L1",
86                  # The root folder where all generated data is going to be saved to
87                  'root_folder'         : "D:/BehaviourData/",
88                  # Timezone information for timestamping
89                  'timezone'            : "US/Mountain",
90                  # Physiological recording made/Technique used. For example: Calciu
91                  'recording_type'      : "Calcium Imaging"
92                  ## End
93                  }
94      #
95      ################################################################################
```

Parameter	Description	Example
inter_trial_interval	The time delay in-between trials.	*5*
decision_interval	The time delay between when an offer is presented and when it is decided if a subject approached the offer.	*3*
feeding_interval	The time delay to allow the subject to consume the reward.	*5*
trials	The number of trials in a session	*40*
levels	A list of the available cost levels in the trials, from lowest cost level at element 0, to the highest cost level at element N-1 (typically N= 4).	*[0, 1, 2, 3]*
feeders	A list of the available feeders representing reward levels in the trials, from the first feeder with the highest reward level at element 0, to the last feeder with the lowest reward level at element N-1 (typically N= 4).	*[1, 2, 3, 4]*
lvl_probs	Probability of a COST level appearing in the trial list. Cost level 0 is represented in element 0, and the highest cost level is represented in element N-1 (typically N= 4). Sum of all elements must equal 1.	*[0, 0.5, 0, 0.5]* *(represents an L1 L3 trial)*
fdr_probs	Probability of a FEEDER appearing in the trial list. Feeder 1 is represented in element 0, and the last feeder is represented in element N-1 (typically N= 4).	*[0.25, 0.25, 0.25, 0.25]*

	Sum of all elements must equal 1.	
rewards	Sugar concentration in each reward solution. If compound solution (for example sucrose + alcohol), include second solute concentration in parenthesis: X%(Y%). No spaces. Serves as a label for the data.	*["9%","5%","2%","0.5%"]*
reward_volume	Reward volume delivered. Does not affect the reward volume delivered by RECORD, serves only as a data label.	*"4ml"*
intensities	Cost intensity, in lux. Including units, no spaces. Does not affect the LED brightness delivered by RECORD, serves only as a data label.	*["0lux","15lux","140lux","290lux"]*
subj_name	Subject's name or ID number, as defined in rat roster.	*"Princess"*
subj_health	Specifies the type of manipulation done. If regular behaviour trial: "Normal"; if alcohol trial: "Alcohol"; if oxy trial: "Oxycodone"; etc...	*"Normal"*
subj_weight	Specify the subject weight, in grams. Include units, no spaces. If no weight is defined, write "Undefined"	*"200 g"*
task_type	Specify the type of task done. For example L1, L2, L3, L1L2, L1L3, etc.	*"L1L3"*
root_folder	The root folder where all generated data is going to be saved to. Can be an absolute or relative path.	*"D:/BehaviourData/"*

timezone	Timezone information for timestamp data. If not defined, defaults to UTC. We use the 'pytz' library to generate timezone data on our 'datetime' timestamps.	*"US/Mountain"*
recording_type	Physiological recording made/Technique used. For example: Calcium Imaging, Optogenetics, Electrophysiology, etc.	*"Calcium Imaging"*

RECORD System Operation

The RECORD system consists of the arenas and its individual components, but of equal importance, the microcontroller unit and the various electronic components that surround it. This section will use PuTTY to demonstrate the operation of the system, but the RECORD system can also be operated via batch script or via Python script using the RECORD-lib package (https://github.com/rjibanezalcala/RECORD/tree/main/python/RECORD-lib).

Available Commands

Command	Description
#	Turns on a specific feeder LED ring at a specified level. The microcontroller will quietly wait for an input and only execute it once four characters have been entered. The second and fourth character dictate the feeder that will be activated and the cost level at which the light will be turned on. Generally, it is recommended that inputs are formatted in the following way: FxLy, where X is the feeder, and Y is the cost level.
F, G, H, and J	Toggles relay 1, 2, 3, and 4, respectively, which will open and close their respective valves. The amount of time that the relay will be closed can be configured in configuration mode.
R	Resets everything but the TTL signals. All LEDs are turned off and relays are opened.
$	Starts configuration mode. Instructions will pop up as soon as you input this command. This will allow you to reconfigure how bright feeder LEDs should be, how long valves stay open, how long TTL pulses are, amongst other settings. *
%	Starts calibration mode. This allows the user to increase and decrease the brightness of the cost LEDs on the fly. It is recommended this command is only used to calibrate the LED brightness of each cost level and not during a trial. *
T	Sends a TTL out through port 3.6. *
K	Toggles the 'trial in progress' light.
Q	Starts the internal timer. This timer counts seconds and milliseconds. Starting the timer will increment the timestamps that show in the acknowledgement message for each command.

W	Gets the current time from the internal timer. Time will be formatted as *[seconds].[milliseconds]*.
E	Stops the internal timer. This will reset the timer back to 0.
Y	Allows the system to respond to external TTLs received through P3.5.
?	Shows system information and a list of available commands with a description of each. *

* All commands (with the exception of 'T', '$', '%', and '?') will be acknowledged with the ACK signal through port 3.0 upon execution

The internal timer

An internal timer can be used to keep track of events that happen throughout a trial. Here, each event refers to each time an event is sent to and acknowledged by the microcontroller. In each acknowledgement message, a timestamp will be reported from the internal timer; if the timer has not been started, the timestamp reported will read "0.0".

Requesting system information

When requesting system information via the '?' command, relevant information will be displayed first, then if a command list is needed, the user can then send an 'H' character. Entering any other character (including a return character) will exit the command.

The microcontroller buttons

The onboard microcontroller buttons serve only as a test to determine if the microcontroller is receiving commands. They currently (firmware v2.2.0) serve no other purpose.

Configuration mode

This mode's main purpose is to set the system up with different operating conditions for a customisable setup, when changes need to be made on the fly. However, it is important to note that any changes made in configuration mode will be lost as soon as the microcontroller is power cycled, as it will boot up with the default parameters set up by the "_cfg.h" file in the microcontroller firmware (https://github.com/rjibanezalcala/RECORD/blob/main/microcontroller/mcucfg_dev1.h). To make changes to the default configuration of the device, the microcontroller must be re-flashed with the desired changes.

When no other commands are active, configuration mode is accessed via the '$' command. It is recommended that this mode be accessed through a command line interface such as PuTTY or Plink, but configuration can be scripted through any scripting language, or through RECORD-lib, our python library (https://github.com/rjibanezalcala/RECORD/tree/main/python/RECORD-lib).

Configuring LED brightness

Configuring the brightness at which the LEDs turn on at each cost level is done by selecting option 'A' after entering configuration mode. The microcontroller will then prompt the user for some input, namely…
1. What feeder needs to be configured,

2. What cost level is to be configured,

3. And the new **CCR value** (see more on CCR values under the "How LED brightness is controlled" section).

Configuring relay active time

How long the relays are held open when using the F, G, H, or J commands is option 'B' after entering configuration mode. How long this is must be expressed in milliseconds as a whole, positive integer number, without exceeding 9000 milliseconds.

Configuring TTL length

Similar to the relay active time, the TTL length parameter cannot exceed 9000 milliseconds and must be expressed in milliseconds as a whole, positive integer number. This parameter is configured in menu item 'C' after entering configuration mode and affects both the user-prompted outgoing TTL ('T' command) and the ACK TTL, which is sent every time a command is executed.

Configuring outgoing TTL operating modes

The user-prompted outgoing TTL ('T' command) can be configured to operate in two different operating modes, and can also be turned off entirely. Selecting menu item 'D' after entering configuration mode will give the user the option to select which operating mode to set.

This TTL is 'low' (logical 0) by default, and will only be set to 'high' (logical 1) when the 'T' command is received, with how it is set varying between operation modes. When high, a non-zero voltage will be observed on the output pin; when 'low', a negligible voltage will appear at the output pin.

Below is a short description of each operation mode…

- *Toggle* is where a TTL is set to high and held at high until the 'T' command is called again.
- *Pulse* is where the TTL is set to high and held only for the amount of time set by the TTL length parameter. When TTL length is elapsed, the TTL will be set back to low.
- *Off* will simply turn outgoing TTL servicing off, meaning that if the 'T' command is received, the microcontroller will only display a message saying that a TTL was requested, but no TTL will be sent.

The outgoing TTL will never be accompanied by an ACK signal, regardless of what operation mode it is operating in.

Incoming TTL servicing

Servicing incoming TTLs may be useful when synchronising the RECORD system with another external system. By default, external TTL servicing is turned off, but can be turned on via the 'Y' command. Upon receiving an external TTL, the Port_3_ISR(void) interrupt service routine (ISR) is called. In version 2.2.0, this ISR does not contain any useful action and must be configured by the user by modifying the firmware; however, actual configuration of the action to be taken when receiving a TTL is a future feature that is under development.
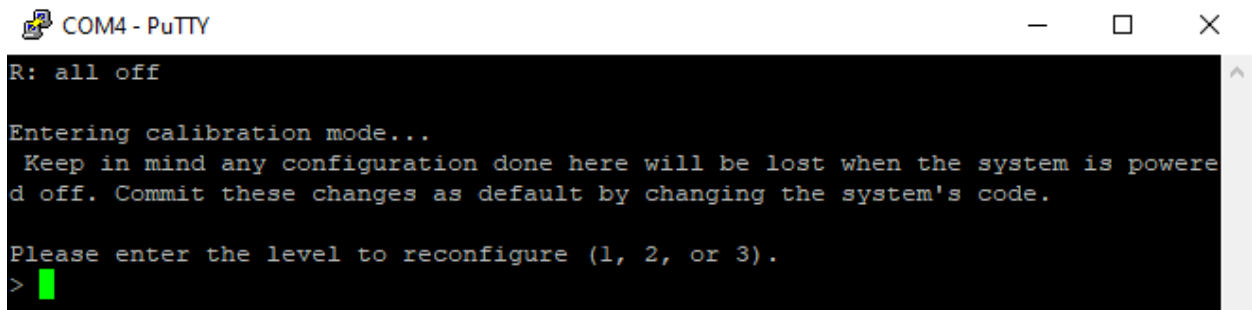
Calibration and Reconfiguration of Cost Levels for the RECORD System

This guide focuses on the reconfiguration and calibration of LED brightnesses for each feeder at all 3 cost levels. Our RECORD system is able to deliver 3 different levels of cost for our decision-making tasks. This is done through pulse width modulation using crystal oscillator-driven timers on a Texas Instruments MSP430 FR2355 microcontroller.

Fine-tuned calibration of cost levels in calibration mode

This section covers calibration mode. In this mode, you are able to increase and decrease the **CCR value** to match a desired **Lux value** without the need of writing a new value each time. Refer to the LED Calibration document for information on what Lux you should aim for. You will need a **lightmeter** on hand to read Lux. *DO NOT reconfigure cost levels unless directed and supervised by a lab supervisor. Misconfigurations can and will affect our data!!*

1. Follow the steps in the previous section to open a communication channel to the microcontroller.
2. Send a '%' (percent symbol) to enter calibration mode. The microcontroller will respond with instructions on how to proceed.
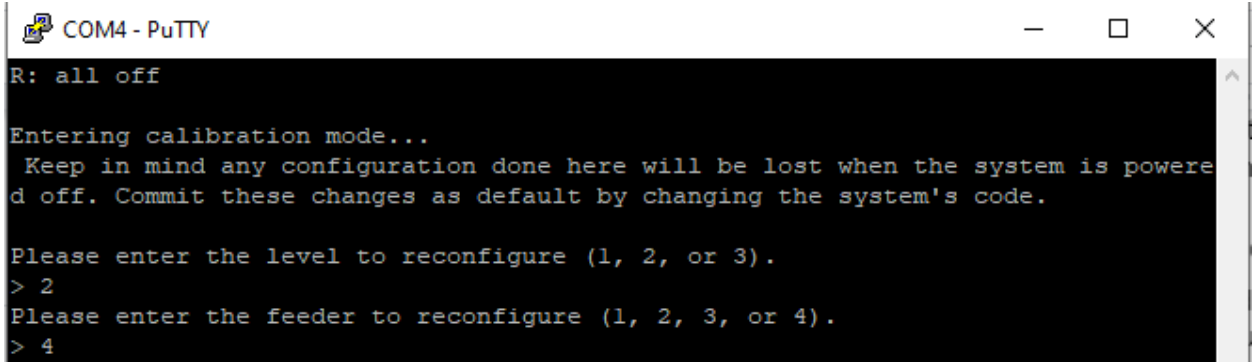


```
R: all off

Entering calibration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
>
```

3. At this point, the microcontroller will ask you for some information, interact with it by using your keyboard. Inputs will automatically be sent without the need to press "Enter" in between inputs. Only one feeder can be calibrated at a specific level at a time.
   a. The first prompt will ask you to specify at which **level** you wish to calibrate a feeder (L1, L2, or L3, see the "What are cost levels?" section for more information). **Enter either 1, 2, or 3 only**, do not enter letters, a number greater than 3, or less than 1, doing so will result in an error message at a later step.
   b. The second prompt will ask you what **feeder** you wish to calibrate at the specified level. **Enter a *whole* number between 1 and 4**, corresponding to each feeder on the arena. Similar to before, do not enter letters, numbers over 4, or numbers below 1:
      i. Entering a 1 will reference the feeder at the **diagonal zone**.
      ii. Entering a 2 will reference the feeder at the **grid zone**.
      iii. Entering a 3 will reference the feeder at the **horizontal zone**.
      iv. Entering a 4 will reference the feeder at the **radial zone**.
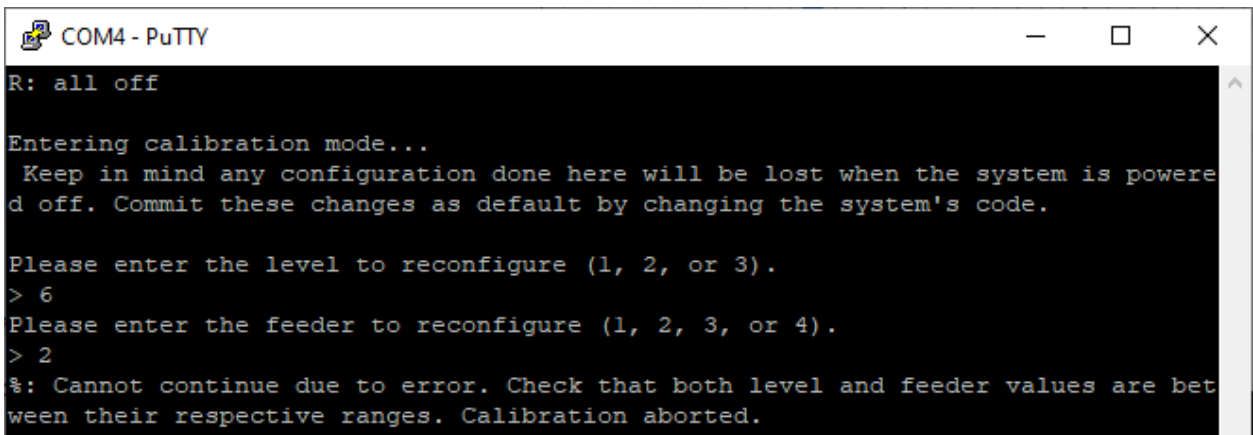
18

```
COM4 - PuTTY                                                    —    □    ×

R: all off

Entering calibration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 2
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 4
```

If an invalid input is entered, an error message will appear and calibration will be aborted. The previously saved parameters will be used until successful reconfiguration.
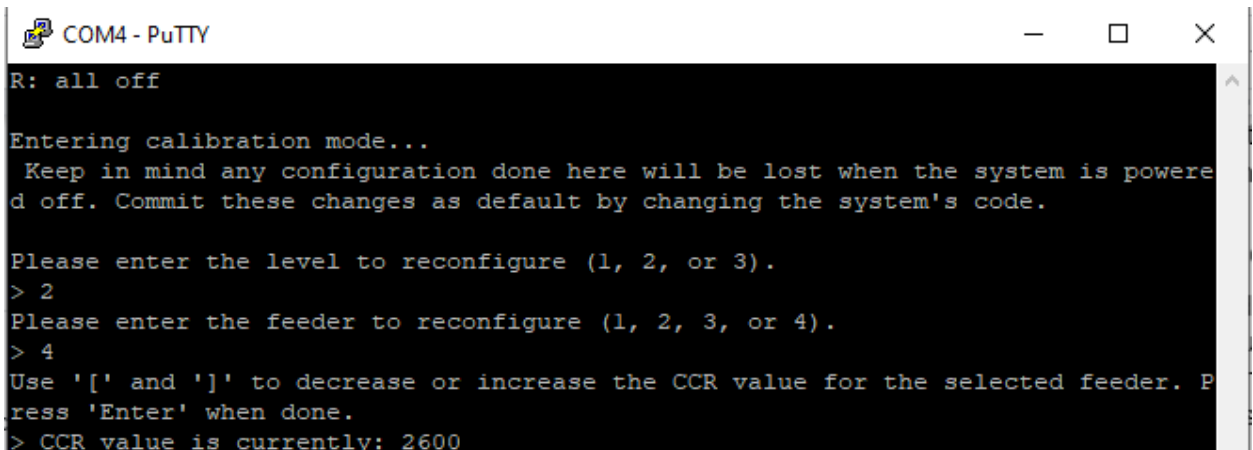
```
COM4 - PuTTY                                                    —    □    ×

R: all off

Entering calibration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 6
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 2
%: Cannot continue due to error. Check that both level and feeder values are bet
ween their respective ranges. Calibration aborted.
```

4. At this point, the feeder you selected should have turned on at the level you're calibrating. The microcontroller will tell you what **CCR value** is currently being used for that level, this will be displayed in PuTTY.

```
COM4 - PuTTY                                                    —    □    ×

R: all off

Entering calibration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 2
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 4
Use '[' and ']' to decrease or increase the CCR value for the selected feeder. P
ress 'Enter' when done.
> CCR value is currently: 2600
```

To increase the brightness on that feeder, one must decrease the **CCR value**, to do this, use the closing square bracket (']') To decrease the brightness by lowering the **CCR value** use the opening square bracket ('['). As before, the **CCR value** will update as you send either symbol, and will be displayed on the PuTTY window.

```
COM4 - PuTTY                                                    —    □    ×

R: all off

Entering calibration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 2
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 4
Use '[' and ']' to decrease or increase the CCR value for the selected feeder. P
ress 'Enter' when done.
> CCR value is currently: 2600
> CCR value is currently: 2550
> CCR value is currently: 2500
> CCR value is currently: 2550
> CCR value is currently: 2600
> CCR value is currently: 2650
> CCR value is currently: 2700
```

Measure the Lux at that feeder using the **lightmeter** and check that the measured value falls within 10 Lux of the target Lux value. **Make sure to write this value down along with the CCR value that produced it!**

5. Once you are satisfied and have written down both the **CCR value** and the corresponding **Lux value**, hit the "Enter" key. The microcontroller will ask you if you wish to save this new **CCR value**. Enter 'y' for **yes**, and 'n' for **no**.

   If you enter 'y', the **CCR value** will be updated and used until the microcontroller is turned off and a message confirming reconfiguration will be displayed and all feeders will be turned on at the level you just calibrated. Only the feeder you were calibrating will be changed, you may be able to tell a difference in brightness.

```
> CCR value is currently: 2750
> CCR value is currently: 2700
> CCR value is currently: 2650
> CCR value is currently: 2600
Would you like to apply these changes? [y/n]
> y
%: New settings applied! Resuming previous operations...
```

   If you enter 'n', all changes from this calibration session will be discarded and the microcontroller will revert back to the last saved **CCR value**. All feeders will turn on.

```
> CCR value is currently: 2850
> CCR value is currently: 2900
> CCR value is currently: 2950
> CCR value is currently: 3000
Would you like to apply these changes? [y/n]
> n
%: Rolled back to previous configuration. Resuming previous operations...
```
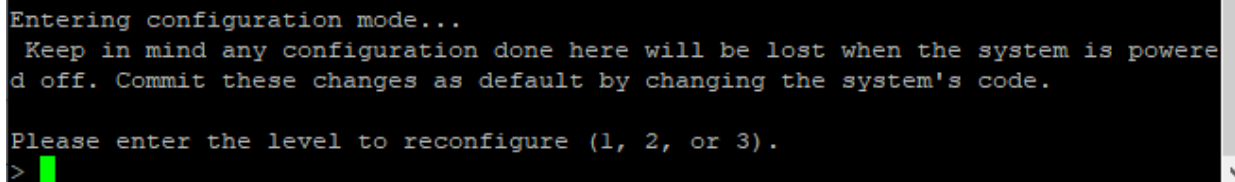
6. After the final message is sent, the microcontroller will go back to its idle state and will be able to receive more commands.

Please only calibrate the LEDs under supervision of the PI or a lab supervisor. Small differences in LED brightnesses can affect our data!!

Reconfiguration of cost levels in configuration mode

This section covers configuration mode. In this mode, you are able to change the **CCR value** directly to match a desired **Lux value**. Refer to the LED Calibration document for information on what Lux you should use. You will need a **lightmeter** on hand to read Lux. *DO NOT reconfigure cost levels unless directed and supervised by a lab supervisor. Misconfigurations can and will affect our data!!*

1. Follow the steps in the opening PuTTY section to open a communication channel to the microcontroller.
2. Send a '$' (dollar sign) to enter configuration mode. The microcontroller will respond with instructions on how to proceed.

```
Entering configuration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
>
```

3. At this point, the microcontroller will ask you for some information, interact with it by using your keyboard. Inputs will automatically be sent without the need to press "Enter" in between inputs. Only one feeder can be calibrated at a specific level at a time.
    a. The first prompt will ask you to specify at which **level** you wish to calibrate a feeder (L1, L2, or L3, see the "What are cost levels?" section for more information). **Enter either 1, 2, or 3 only**, do not enter letters, a number greater than 3, or less than 1, doing so will result in an error message at a later step.
    b. The second prompt will ask you what **feeder** you wish to calibrate at the specified level. **Enter a *whole* number between 1 and 4**, corresponding to each feeder on the arena. Similar to before, do not enter letters, numbers over 4, or numbers below 1:
        i. Entering a 1 will reference the feeder at the **diagonal zone**.
        ii. Entering a 2 will reference the feeder at the **grid zone**.
        iii. Entering a 3 will reference the feeder at the **horizontal zone**.
        iv. Entering a 4 will reference the feeder at the **radial zone**.

```
Entering configuration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 3
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 1
Please enter the new integer CCR value for this level and feeder (0 through 8000
, whole numbers only).
Enter a 4-character number or press enter if you're entering less than 4 charact
ers.
>
```

4. The microcontroller will ask for a new **CCR value** for the level and feeder you are reconfiguring. Enter a *whole* number between 0 and 8000 *only*. If your new **CCR value** consists of three digits or less, press the "Enter" key to submit the new value.

    Please refer to the LED Calibration *spreadsheet for information about what values result in what Lux. Only use values contained in this spreadsheet unless you are calibrating a feeder!*

```
Entering configuration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 3
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 1
Please enter the new integer CCR value for this level and feeder (0 through 8000
, whole numbers only).
Enter a 4-character number or press enter if you're entering less than 4 charact
ers.
> 400
```

5. After entering four digits or pressing the "Enter" key, the microcontroller will ask you if you would like to preview the new brightness. Enter 'y' for **yes**, and 'n' for **no**.

```
Please enter the new integer CCR value for this level and feeder (0 through 8000
, whole numbers only).
Enter a 4-character number or press enter if you're entering less than 4 charact
ers.
> 400
Would you like to test the new value? [y/n]
>
```

    If you enter 'y', all feeders will turn on at the level you have just reconfigured, however only the feeder you specified in step 3 will be affected. Changes will be applied and saved for the rest of the session (until the microcontroller is turned off).

```
Enter a 4-character number or press enter if you're entering less than 4 charact
ers.
> 400
Would you like to test the new value? [y/n]
> y
$: New settings applied! Resuming previous operations...
```

If you enter an 'n', the new **CCR value** will still be applied, but the feeder will need to be turned off and then back on again for the changes to become evident.

```
Enter a 4-character number or press enter if you're entering less than 4 charact
ers.
> 400
Would you like to test the new value? [y/n]
> n
$: Configuration applied! Restart feeders to see changes. Resuming previous oper
ations...
```

If an invalid input is entered, an error message after the new CCR value is entered will appear and calibration will be aborted. The previously saved parameters will be used until successful reconfiguration.

```
Entering configuration mode...
 Keep in mind any configuration done here will be lost when the system is powere
d off. Commit these changes as default by changing the system's code.

Please enter the level to reconfigure (1, 2, or 3).
> 5
Please enter the feeder to reconfigure (1, 2, 3, or 4).
> 8
Please enter the new integer CCR value for this level and feeder (0 through 8000
, whole numbers only).
Enter a 4-character number or press enter if you're entering less than 4 charact
ers.
> 777
Error: New CCR value should not exceed 8000 or LEVEL should not exceed 3. Config
uration not set.
$: Configuration aborted due to error, try again. Resuming previous operations..
```

6. Measure the brightness of the feeder using a **lightmeter**, then record the new **CCR value** and its corresponding **Lux value**. After the final message is sent, the microcontroller will go back to its idle state and will be able to receive more commands.

Please only calibrate the LEDs under supervision of the PI or a lab supervisor. Small differences in LED brightnesses can affect our data!!

Suggested procedure to calibrate LEDs

Relevant documentation

Refer to the following documentation when calibrating the RECORD system feeder LEDs:
1. LED Calibration Data:
   a. Arena 1 Calibration (Sheet)
   b. Arena 2 Calibration (Sheet)
   c. Arena 3 Calibration (Sheet)
   d. Arena 4 Calibration (Sheet)
   e. Per-Rat Lux Requirements (Sheet)
2. Reconfiguration of cost levels in configuration mode
3. Fine-tuned calibration of cost levels in calibration mode

Note: DO NOT edit these documents unless otherwise instructed to by the PI or a lab supervisor!

Creating LED lines of best fit to aid in calibration

The "**calibration lines**" or "**trend lines**", help us estimate a CCR value that may possibly produce the appropriate amount of Lux we are looking for. These are calculated from "**calibration curves**", which we create by measuring Lux at 6 different CCR values. The **calibration line** is the trend these curves follow, also known as the "line of best fit" or "regression line", which results from a statistical calculation called the "Least Square Method". Our LED Calibration Data document does this automatically, all that needs to be done is enter the 6 measurements mentioned before. The following steps will guide you on how to do this, *you will need a **lightmeter** able to measure up to 1000 Lux to proceed*.

1. Open the LED Calibration Data document and open one of the Arena Calibration sheets. You will be met with a small table with numbers, underneath it will be a large graph with four coloured lines, these are the **calibration curves**.
2. Check with a lab supervisor or the PI to know if the data is accurate; the date of last measurement is located to the right of the first table on cell H2, the data may or may not need to be updated, depending on changes made to the arenas. If an update is needed, proceed to step 3.

eder 4 Radi)	Last measurement date:	25/March/2022
0		

3. Open PuTTY and connect to the microcontroller for the arena you are calibrating, see the "Opening PuTTY and connecting to an arena microcontroller" section on this guide for instructions on how to do so.
4. Using the steps in the "Reconfiguration of cost levels in configuration mode" to enter **configuration mode** on the microcontroller, start by configuring the microcontroller to turn all the feeders on to a CCR value of either 6400, 4800, 3200, 1600, or 1. You may choose any level for this calibration when prompted.
5. Turn on the **lightmeter** and make sure to set it to measure Lux, not Fc or Lumen. Then, set the measurement range to 0 - 2000 Lux by pressing the "R" button until the range indicator on the bottom on the **lightmeter** screen underlines the 2000 Lux range, you may be able to measure using the lowest range if you are calibrating cost level 1, but if the **lightmeter** reads "OL" at any point, you will need to step up a range. Finally, because the measured value will constantly be changing, it helps to take only the maximum value. Point the **light sensor** (it looks like a white half-sphere) and set the **lightmeter** to "Max" mode then at a dark spot on the arena, this will freeze the measured value.
6. Start by measuring feeder 1 on the horizontal corner of the arena. Take the **light sensor** and hold it about 3 millimetres or less above each one of the four LEDs on the feeder, holding it in place for 2 to 3 seconds. The lightmeter will display the maximum value it was able to measure. It is recommended you measure each feeder at least twice, as there will be a wide variability in the measurement.
7. Record the values you measured for each feeder into the appropriate row for the CCR value you're measuring, you'll start to see the calibration curve graph change.
8. Once you are done with that CCR value, move on to the next and repeat steps 4 through 7 until you've measured all feeders at each CCR value.
9. The **trend line equation** will appear on the calibration curve graph in the legend on the right side.

<div align="center">

━ Feeder 1 (Diag)
━ -0.0704*x + 553 R² = 0.998
━ Feeder 2 (Grid)
━ -0.0275*x + 217 R² = 0.998
━ Feeder 3 (Hori)
━ -0.032*x + 249 R² = 0.981
━ Feeder 4 (Radi)
━ -0.0356*x + 278 R² = 0.995

</div>

X in this equation is the CCR value, and Y is the Lux. R^2 is the correlation coefficient, the higher this number is, the better the line fits the data, you do not need to worry about this value for now. Let's generalise this equation to $Y = A_{fdr}*X + B_{fdr}$, where A_{fdr} and B_{fdr} are the corresponding slope and Y-intercept values for each feeder's **trend line**. This is the first and last bit of maths you'll need to know for this guide.

10. Solve the equations for X. Below is an example using the first equation marked in muted blue:

$$Y = -0.0704\ X + 553$$
$$Y - 553 = -0.0704\ X + 553 - 553$$
$$Y - 553 / (-0.0704) = (-0.0704\ X) / -0.0704$$
$$\mathbf{Y - 553 / (-0.0704) = X}$$

Set up this equation in terms of spreadsheet cells for the **CCR value estimator**:

    a. For feeder 1, enter "=ROUND(($R33$-$\mathbf{B_{fdr}}$)/$\mathbf{A_{fdr}}$)" into cell R34
    b. For feeder 2, enter "=ROUND(($R36$-$\mathbf{B_{fdr}}$)/$\mathbf{A_{fdr}}$)" into cell R37
    c. For feeder 3, enter "=ROUND(($R39$-$\mathbf{B_{fdr}}$)/$\mathbf{A_{fdr}}$)" into cell R40
    d. For feeder 4, enter "=ROUND(($R42$-$\mathbf{B_{fdr}}$)/$\mathbf{A_{fdr}}$)" into cell R43

11. Once done setting up the **CCR value estimator**, you are done. Notify a lab member to double check your work!

Calibrating feeder LEDs to rat-specific required Lux values

It is important to know where to start with the calibration process. We have set up a chart with **target lux values**, individualised for each rat in the Per-Rat Lux Requirements sheet on the LED Calibration Data documentation. This document tells you what Lux value to aim for at each cost level when calibrating the RECORD system; but how do you know what CCR value will give you these values?

In the same document, there are four sheets that describe the LED brightness behaviour at each arena, we call these "**calibration lines**" or "**trend lines**". These graphs are carefully created by our team to predict a ballpark estimate of the CCR value needed for the desired Lux values. The sheets also contain a history of CCR values that were found to produce a Lux value within the desired ranges, in case we need to dial back or crank up the brightness without putting in more effort than needed. See the "Creating LED calibration lines to aid in calibration" section in this guide to learn how these calibration lines are obtained.

Follow the following steps to begin calibrating, *you will need a **lightmeter** able to measure up to 1000 Lux to proceed*:

1. Open the Per-Rat Lux Requirements sheet and refer to columns E, F, and G to find out what Lux value you are to aim for. Choose a level, then choose a Lux value to begin with, this will be your **target Lux value**.

Index	Rat Name	Sex	ID No.	Level 1 Target Lux	Level 2 Target Lux	Level 3 Target Lux		
1	Alexis	F	14290.1	15.0 Lux	40.0 Lux	240.0 Lux	Last Updated: 08/April/2022	
2	Sarah	F	14290.2	15.0 Lux	40.0 Lux	240.0 Lux		
3	Kryssia	F	14291.1	15.0 Lux	40.0 Lux	240.0 Lux	Level Probability of Appearance (L1/L2/L3):	(70 / 0 / 30)%
4	Raven	F	14291.2	15.0 Lux	40.0 Lux	240.0 Lux		
5	Harley Quinn	F	14292.1	15.0 Lux	40.0 Lux	240.0 Lux		
6	Shakira	F	14292.2	15.0 Lux	40.0 Lux	240.0 Lux		
7	Raissa	F	14293.1	15.0 Lux	40.0 Lux	240.0 Lux		
8	Renata	F	14293.2	15.0 Lux	40.0 Lux	240.0 Lux		
9	Andrea	F	14294.1	15.0 Lux	40.0 Lux	240.0 Lux		
10	Neftali	F	14294.2	15.0 Lux	40.0 Lux	240.0 Lux		
11	Fiona	F	14295.1	15.0 Lux	40.0 Lux	240.0 Lux		
12	Juana	F	14295.2	15.0 Lux	40.0 Lux	240.0 Lux		
13	Sully	M	14296.1	15.0 Lux	40.0 Lux	180.0 Lux		
14	Mike	M	14296.2	15.0 Lux	40.0 Lux	180.0 Lux		
15	Jafar	M	14297.1	15.0 Lux	40.0 Lux	180.0 Lux		
16	Aladdin	M	14297.2	15.0 Lux	40.0 Lux	180.0 Lux		
17	Kobe	M	14298.1	15.0 Lux	40.0 Lux	180.0 Lux		
18	MJ	M	14298.2	15.0 Lux	40.0 Lux	180.0 Lux		
19	Junior	M	14299.1	15.0 Lux	40.0 Lux	180.0 Lux		
20	Carl	M	14299.2	15.0 Lux	40.0 Lux	180.0 Lux		
21	Scar	M	14300.1	15.0 Lux	40.0 Lux	180.0 Lux		
22	Simba	M	14300.2	15.0 Lux	40.0 Lux	180.0 Lux		
23	Jimi	M	14301.1	15.0 Lux	40.0 Lux	180.0 Lux		
24	Johnny	M	14301.2	15.0 Lux	40.0 Lux	180.0 Lux		

2. Choose an arena to calibrate and check with a lab supervisor that the calibration lines are accurate to the feeders currently present in the arena. If the calibration lines are inaccurate, move on to a different arena.

3. Enter your **target Lux value** in the "Desired Lux" fields for each of the four feeders on the **CCR value predictor** (cells R33, R36, R39, and R42). The predictor will calculate a rough estimate of a CCR value you can start with.

Feeder 1 (Diagonal, Blue)		Level 1	Level 2	Level 3	Level 3	Level 3
Desired Lux =	180	15		142		239
Calculated CCR =	5298	7742		5925		4600
Feeder 2 (Grid, Red)						
Desired Lux =	240	16		142		241
Calculated CCR =	-836	7800		5950		4100
Feeder 3 (Horizontal, Yellow)						
Desired Lux =	240	15		137		240
Calculated CCR =	281	7700		5750		3281
Feeder 4 (Radial, Green)						
Desired Lux =	240	14		141		243
Calculated CCR =	1067	7388		3876		868
	Target Lux:	15	40	140	180	240
	Calibration last updated on:	02/April/2022				

Note: If the calculated CCR value happens to be negative, continue with a value of 0. If it is over 8000, proceed with a value of 8000. Do not use any values lower than 0, or above 8000, as this may result in unexpected microcontroller behaviour.

4. Open PuTTY and connect to the microcontroller for the arena you are calibrating, see the "Opening PuTTY and connecting to an arena microcontroller" section on this guide for instructions on how to do so.

5. With the **estimated CCR values** you got, enter **configuration mode** on the microcontroller by sending a dollar sign ('$') to the microcontroller.

6. Follow the steps in the "Reconfiguration of cost levels in configuration mode" to enter the **estimated CCR values** to the microcontroller. You may or may not be able to see the lights change in brightness. Don't worry if you can't, we're making very small changes to the brightness here.

7. Turn on the **lightmeter** and make sure to set it to measure Lux, not Fc or Lumen. Then, set the measurement range to 0 - 2000 Lux by pressing the "R" button until the range indicator on the bottom on the **lightmeter** screen underlines the 2000 Lux range, you may be able to measure using the lowest range if you are calibrating cost level 1, but if the **lightmeter** reads "OL" at any point, you will need to step up a range. Finally, because the measured value will constantly be changing, it helps to take only the maximum value. Point the **light sensor** (it looks like a white half-sphere) and set the **lightmeter** to "Max" mode then at a dark spot on the arena, this will freeze the measured value.

8. Start by measuring feeder 1 on the horizontal corner of the arena. Take the **light sensor** and hold it about 3 millimetres or less above each one of the four LEDs on the feeder, holding it in place for 2 to 3 seconds. The lightmeter will display the maximum value it was able to measure. It is recommended you measure each feeder at least twice, as there will be a wide variability in the measurement.
    a. If the value you measured is within ±10 Lux of the **target Lux**, record this value, the CCR value that produced it, and which feeder on which maze you were calibrating.
    b. If the value is *not* within ±10 Lux of the **target Lux**, use **calibration mode** on the microcontroller by sending a percent sign ('%') to finetune the CCR value and repeat steps 7 and 8 until you reach the target. For instructions on how to operate in **calibration mode**, see the Fine-tuned calibration of cost levels in calibration mode section in this guide. Once you have reached your target, save the new CCR value, and record it like you would in (a).

9. Move on to the next feeder. Once you are done, double check your data and hand it to a lab supervisor so that they may database it.

# How to upload/update Firmware on a Texas Instrument Microcontroller Dev. Board

This guide assumes the user has an existing workspace and project for their microcontroller development board and is specific to Texas Instrument products. Please ensure you are using a Texas Instruments microcontroller development board.

Before starting, disconnect all microcontrollers from the PC and leave only the one to be updated connected. Make sure the microcontroller is on and functioning correctly by looking for a green light on the top of the microcontroller. If everything looks fine, proceed.

1. Start by opening Code Composer Studio. The software version shown in this guide is Version: 10.2.0.00009.
2. Drop down your project hierarchy on the Project Explorer panel on the left side of the screen.
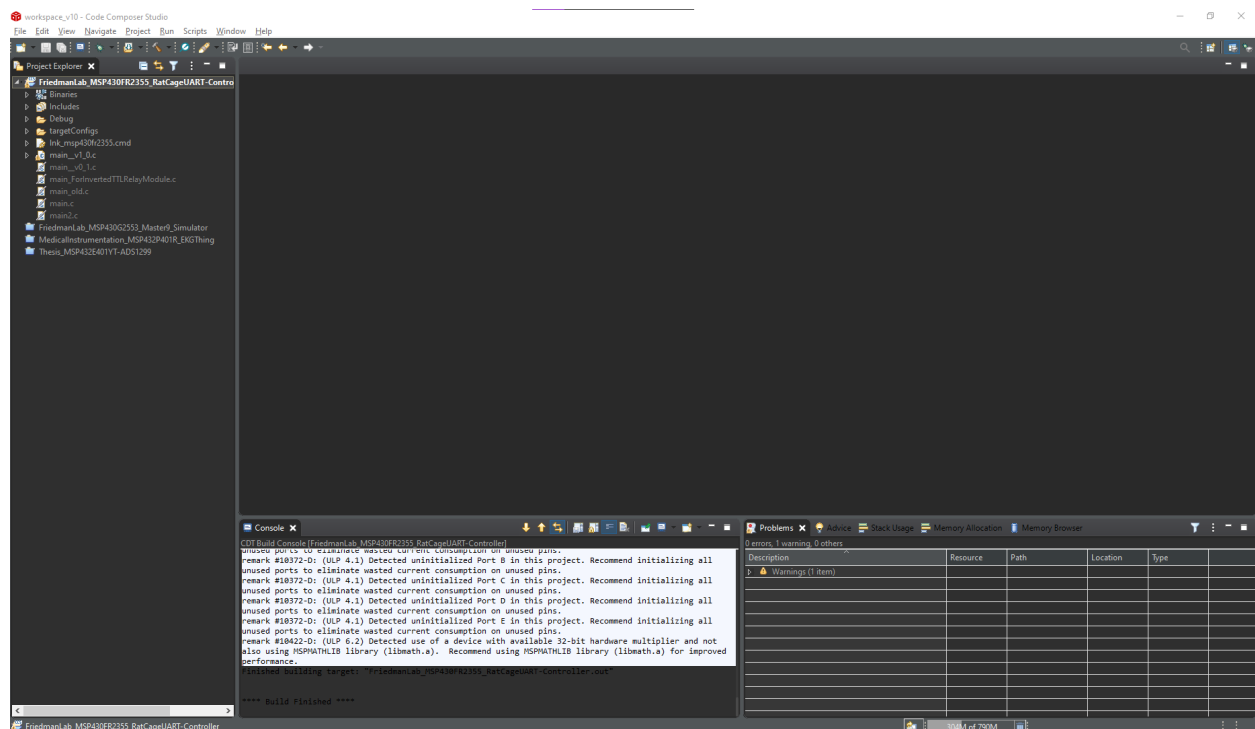
3. Right-click on the top of the hierarchy, then navigate to "Show in Local Terminal". Click on "System Explorer" to open the folder where your entire workspace is saved.
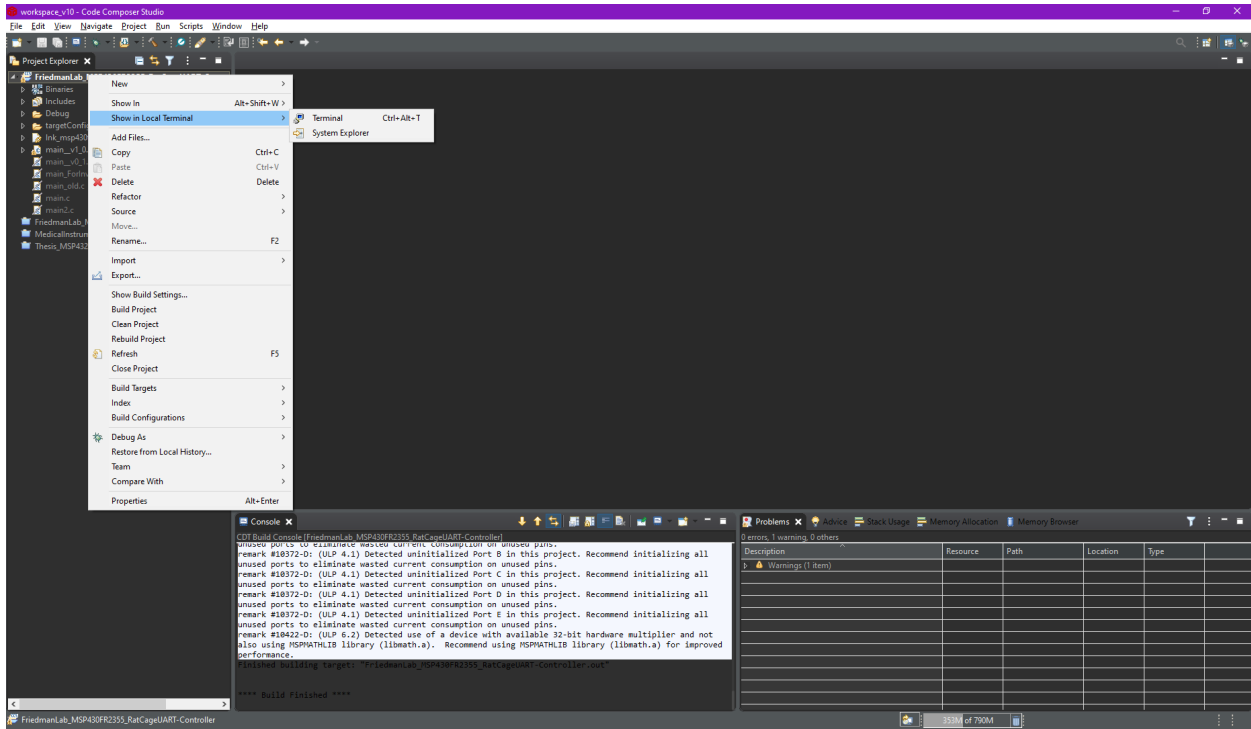
4. Navigate to the project where you wish to upload the new firmware. You will either see several .c files which contain code for the microcontroller, or you will not see any .c files.

5. Paste the new main.c file which contains the new firmware into this folder. In this case we will be updating from main__v0_1.c to main__v1_0.c. Close the file explorer window once you are done.

6. Go back into Code Composer. The new file should appear under your project hierarchy. Right click on the old firmware file and select "Exclude from Build". This will make the program ignore this file when uploading code to the microcontroller.
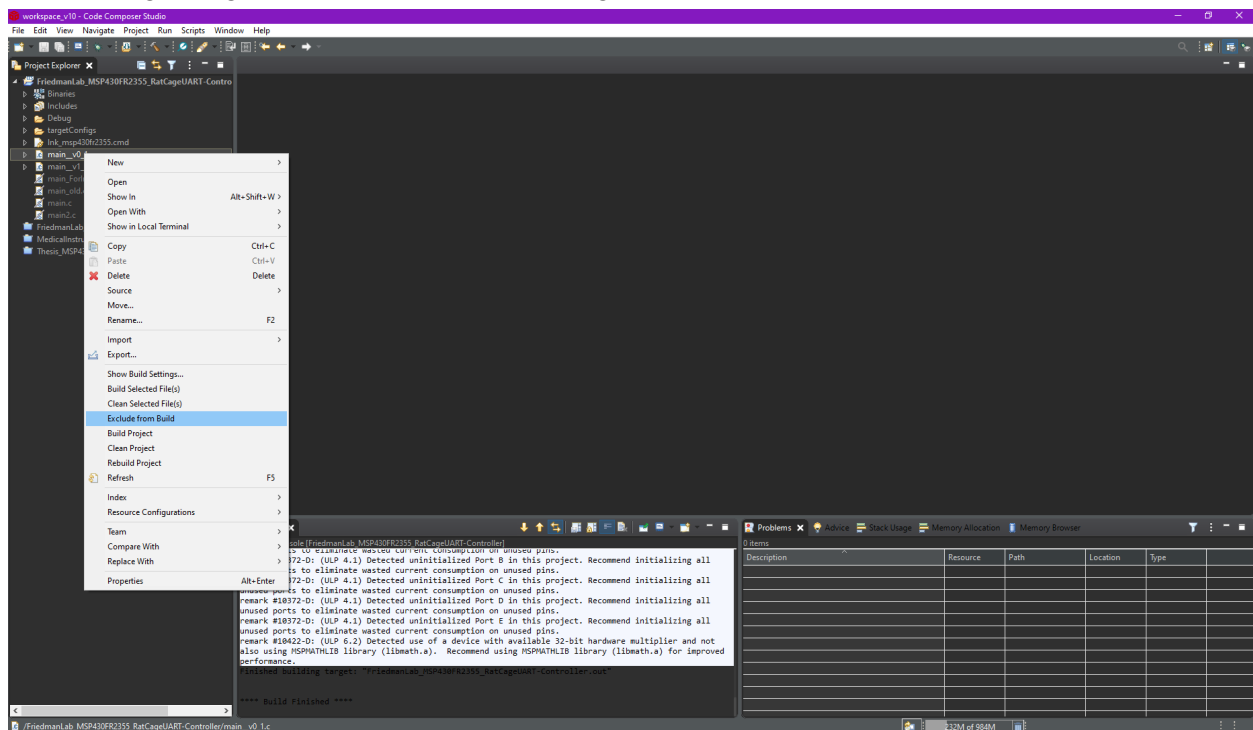

7. Double click on the new file to open it. Code should now appear on your screen. Click on the hammer icon which appears on the toolbar on the top. This will build the project and

check the new code for errors. Text will appear on the console located on the bottom of the screen, simply wait for it to finish. The first build may take longer than subsequent ones.
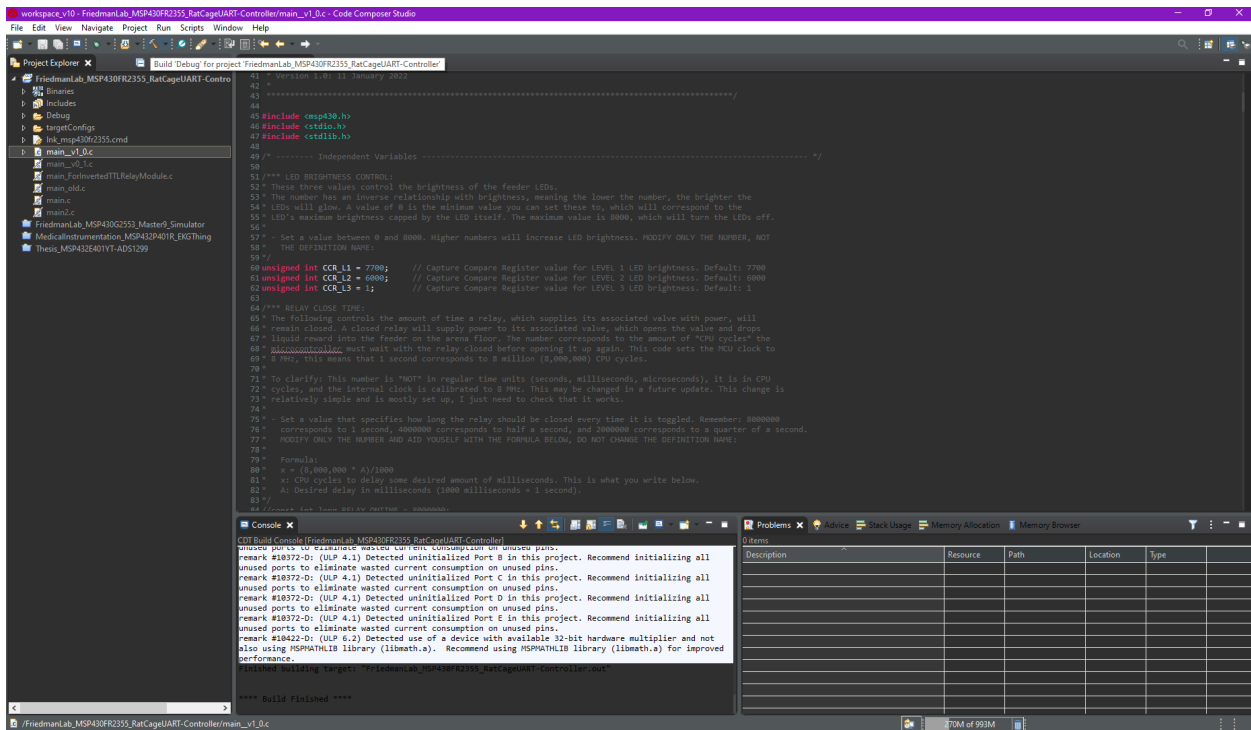
8. Click on the Flash icon located to the left of the hammer icon. This will upload the code to the microcontroller. Click okay if the program asks you to Save and Launch and then click Proceed on the next prompt.
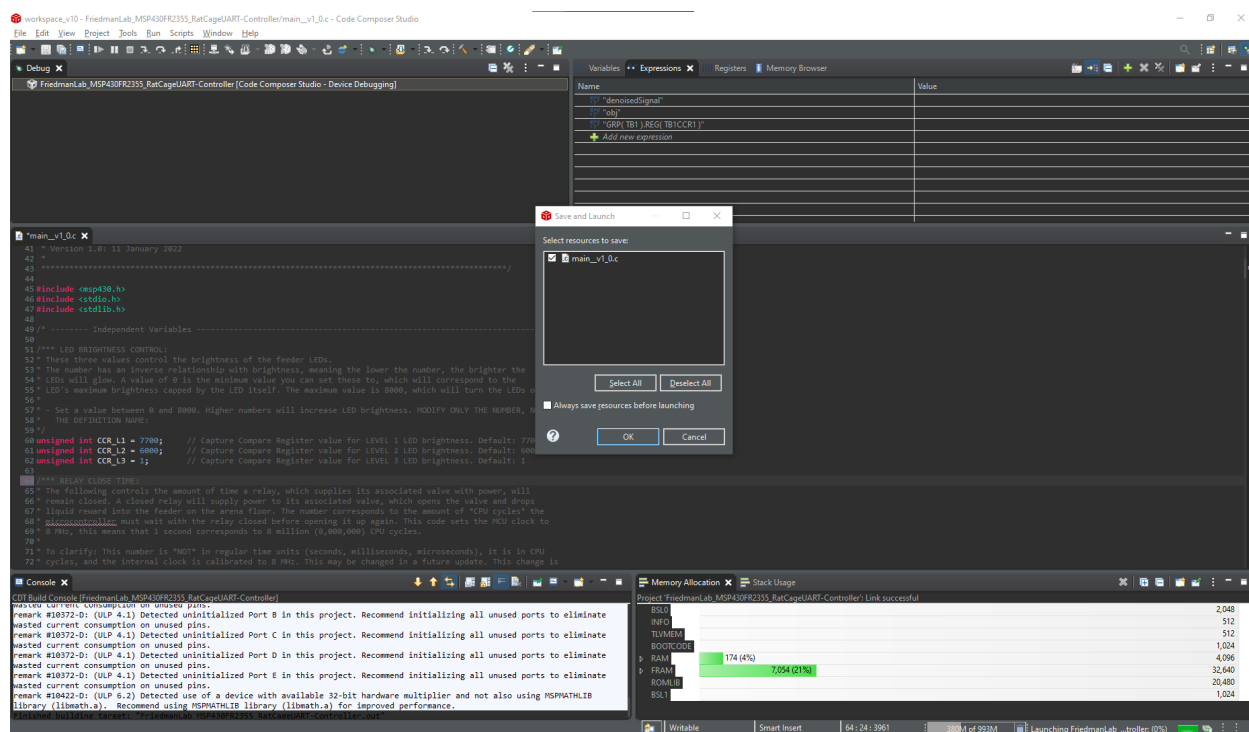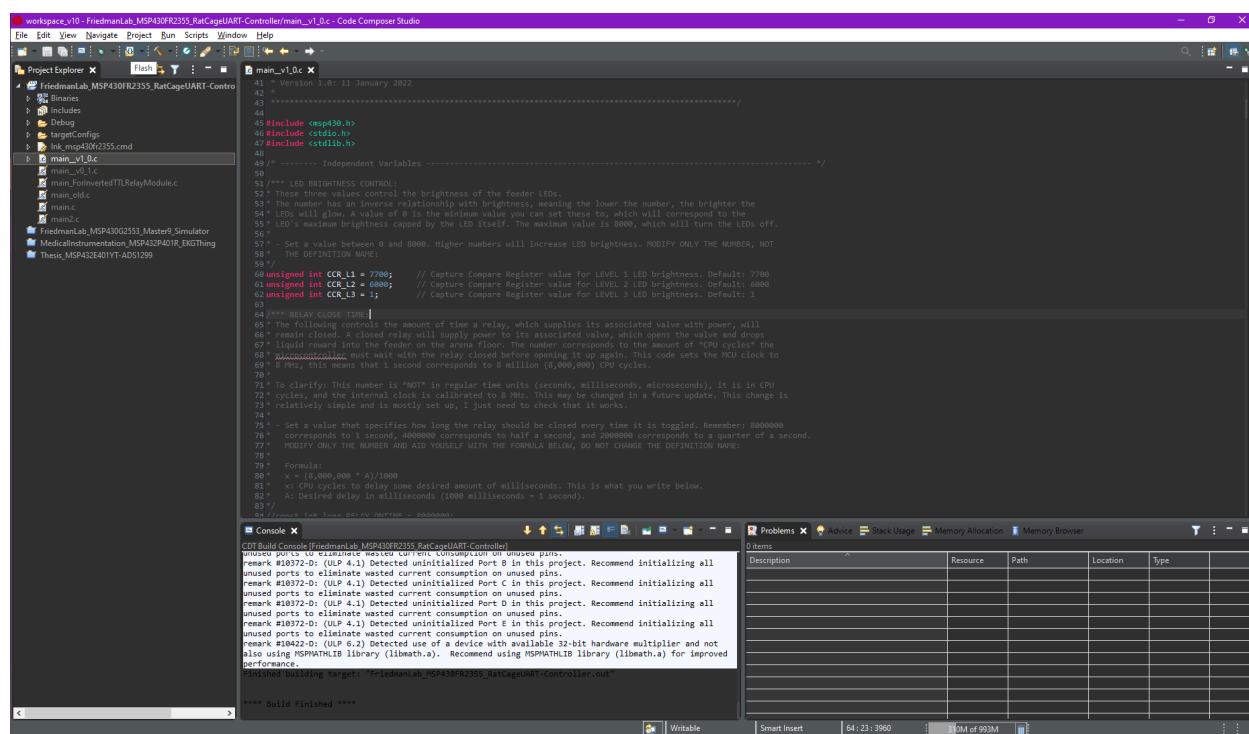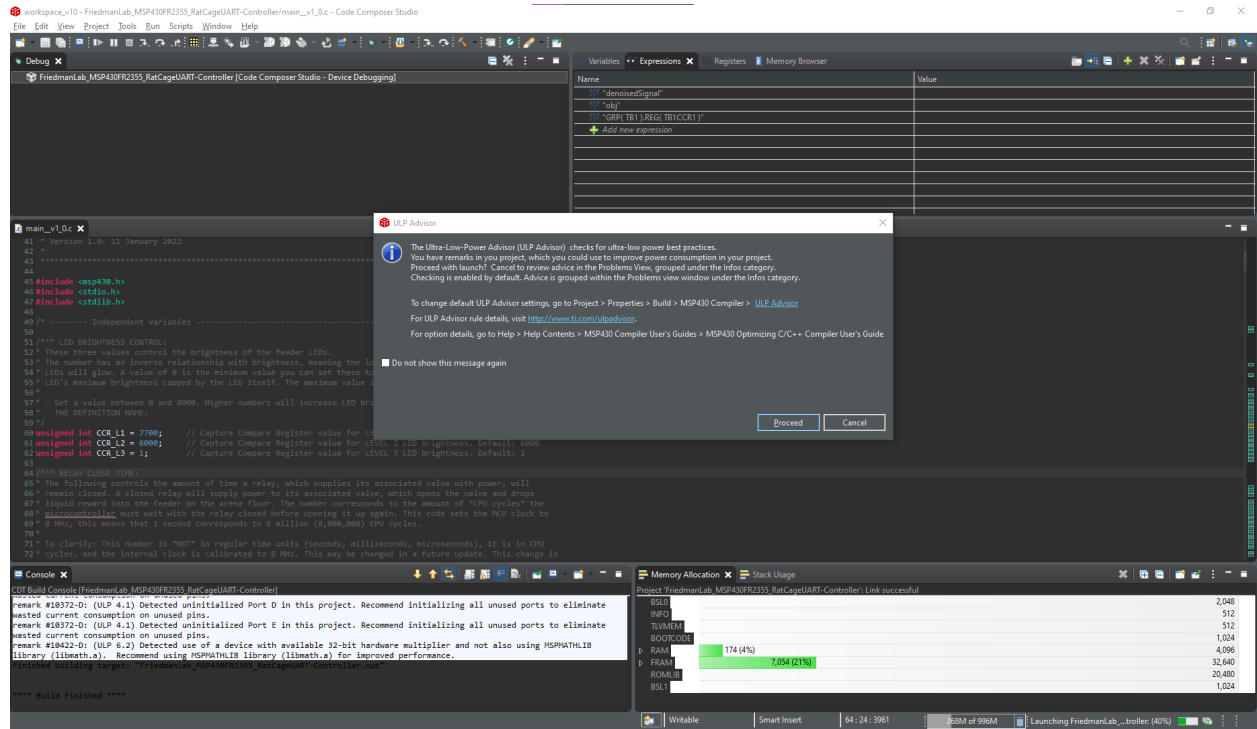
9. The code has now been uploaded to the microcontroller. You may begin using it immediately.

Sometimes, new firmware may be buggy and one may need to roll back an update. To do this, repeat step 6 to re-include the old firmware code in the build, and instead exclude the new firmware code from the build. Then continue on to the next steps.

Troubleshooting

By Graham Waller and Raquel Ibáñez Alcalá

Hardware troubleshooting

Anything concerning the physical components of the RECORD electronics and arenas will be contained here. When in doubt, turn everything off and on → unplug and plug. Make sure everything, from the wires and cables, to the tubes that deliver sucrose, is connected.

Valves and Relays

Problem: Relay clicking, but valves not clicking…
Solutions: Before proceeding, check that the valve cable is connected to a valve on one side, and to the corresponding slot on the RECORD box or circuit board on the other side.Use PuTTY to activate each valve individually by sending "F" for valve 1, "G" for 2, "H" for 3, then "J" for 4, and identify which valve clicks and which does not. See ***Protocol for using PuTTY*** on slack in #important-documents.

If you have at least one valve that does click, swap the connectors for a working valve and a non-working valve then use PuTTY to activate both valves.
   a. If the non-working valve clicks and the working valve does not, the problem may be at the valve cable. Proceed to step (1) below.
   b. If the working valve clicks and the non-working valve does not, the valves may be the problem. Jump to step (3) below.

If no valves click, make sure that the valves are being supplied the correct voltage for them to operate (15 V - 24 V). Check that the power supply cable is connected to the RECORD box and the valve power supply is turned on, then test again. If the problem persists, proceed to step (1) below.

   1. Check connections and cables for short circuits. Make sure all exposed copper wire is shielded; wires should not be touching neighbouring wires at the head of connections to the PCB or the valves. *If needed, tape each individual section of exposed wire with electrical tape to shield connections from one another.*
   2. Check the connector at the valve side of the valve cable, it is possible that the wire may have come loose from its housing and might not be making contact with the connector on the valve. Alternatively, the contacts on the connectors may be dirty. *Disconnect the valve from the cable and clean the contacts using a damp paper towel, then connect them again.*

3. If the problem persists, it may be necessary to either clean or replace the non-working valve(s). To clean the valve, dismount it from the valve stand, remove the solenoid from it, and submerge the valve plunger in water for at least 1 hour, then test it again.

Problem: Relays not clicking…
Solution: Likely caused by a connection issue between the microcontroller and relays. Carefully open the RECORD box and look for the wires that connect the microcontroller (in red) and the relays (long blue board featuring several blue blocks with white writing). If these wires are disconnected, the relays will not click. Refer to the ***Electronics build guide*** to repair and ask a lab supervisor for help, **do not connect or disconnect anything on the PCB or microcontroller without supervision.**

Problem: Valves and relays are clicking, but no sucrose is being dispensed…
Solution: This is likely caused by air being trapped in the system. Before proceeding, check that all tubing is connected to its corresponding feeders. Check for air bubbles in the tubing between the syringes and the valves, and between the valves and the arena feeders. If there are bubbles present, activate the valves a few times using PuTTY, or run the *system_cleanup* batch script routine. You may also force sucrose solution into the system by using the syringe plunger to push down the solution while the valves are open. To do this, take a syringe plunger and while activating the valve with PuTTY, push the plunger down the syringe to flush liquid through and clear the blockage. When finished, check for leaks in the system and fix them accordingly.

———————————————————————

Lights

Problem: Light does not turn on for one or more feeders...
Solutions:
1. Make sure that the feeder lights are active at any (visible) brightness. Use PuTTY to turn the lights on. See ***Protocol for using PuTTY*** on slack in #important-documents.
2. Check that the non-working feeder lights are connected to the LED cable, this should be a two-wire connector that attaches to the cable. If the feeder is connected, make sure the lights are not connected backwards. Some feeders will have a blue and black wire pair, others will have a yellow and green wire pair, and (though uncommon) others may have a red and black wire pair. Make sure that the black wire on the LED cable (representing ground) is connected either to the black or green wire on the feeder LEDs, and the red wire on the LED cable (representing live wire) is connected to the yellow, red, or blue wire on the feeder LEDs. The light should turn on after this.
3. If the light does not turn on, or if one some LEDs turn on on that feeder, lift the feeder up by pushing the "tail" of the feeder up (where the tubing attaches) and pull it up and off the feeder base. Flip the feeder over and look at the underside. *Carefully* separate any exposed metal contacts on the LED ring bundle, making sure there are no short-circuits in the loop. Handle these wires with care, making sure not to break any of the connections. The lights should turn on after separating all the wires.

4. If the lights still do not turn on, check connections and cables for short circuits. Make sure all exposed copper wire is shielded; wires should not be touching neighbouring wires at the head of connections to the PCB or the valves. *If needed, tape each individual section of exposed wire with electrical tape to shield connections from one another.*

Problem: Two lights turning on at once…
Solution: The given cable's copper wires are likely touching at the white connector on the end of the cable which connects to the RECORD box. Use electrical tape to shield each individual exposed copper wire to prevent the wires from touching one another.

Arena Pieces

Pieces will wear down and come apart over time, this process can happen a lot sooner if the rats gnaw on the arena. Simply replace faulty pieces with new parts and/or glue them back together.


Ethovision troubleshooting

Good rule of thumb to avoid issues in EthoVision **is to set the program priority** for EthoVision to *"Real Time"* via TaskManager (ctrl → shift → esc) and having other programs (especially big ones like MatLab and Google Chrome) closed while running trials.

Problem: Trials getting stuck at the beginning stage...
Alternative problem 1: Trial never ends…
Alternative problem 2: Lights stay on after a trial ends…
Solution: Caused by desynchronization between the RECORD system and the animal tracking software. *Stop the trials.* Close all command windows that may have been left open, then turn the microcontrollers off, wait up to 10 seconds, and turn them back on. *Do this only after stopping the trials, doing so during a trial will desynchronize both systems again!*
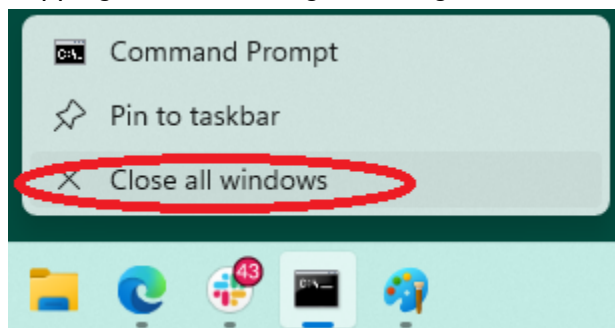
If the problem persists, check that the microcontroller is connected to the I/O interface via one of the data lines on the hardware synchronisation cable. It is possible that the connection might need to be flipped around so that the output from the microcontroller goes to the TTL input line, and the input for the microcontroller goes to the TTL output line.

Changes to the trial control may be needed if the problem persists as the trial may not be allowing enough time for the microcontroller to resynchronize with the trial after running a command, or the serial communication channel may be closing prematurely.

Problem: EthoVision crashed and data was not recovered…
Solution: This is more of a workaround than a solution. Create a backup of the experiment, then create a new experiment and delete any other behavioural task trial control structures from the one which you wish to run. A good practice is to keep different task versions in separate experiments, and always back up your experiments. This is to keep bugs in the program isolated to only one experiment.

Problem: A feeder has a significant delay before delivering sucrose in one or more arenas…
Solution: Check configuration for zones under the trial control settings, make sure that options for each decision-making condition variable (box x accepts, box y rejects, etc.) are set to the appropriate settings. Also increase the time for which accept/reject conditions must be met (for example 0.5s might be too low, and should be increased to 1.0s).
  Example: We incurred this error when the condition check in "box 4 rejects" was set to "when centre-point is in all zones simultaneously" instead of "in any of the zones" in the trial control settings. This made the trial hang because that condition can never be met as the rat cannot exist in more than one zone simultaneously, and the program was waiting until the "box 4 accepts" condition was met.

Problem: The Windows "USB disconnect" sound plays when more than one valve/relay clicks, then the current trial never ends…
Solution: This may be due to a power supply overload in the system, which makes the microcontrollers reset and de-synchronizes them from EthoVision.

If this happens to you in the middle of running trials, manually stop the trial, close all command windows, and continue running trials.

To prevent this problem from happening again, modify the trial control in EthoVision so that the delivery stage at each arena is delayed 0.5 seconds after the previous one, this will ensure that valves don't click at the same time and that the power supply for the relays is not overloaded.

Problem: Ethovision gives an error saying calibration lines are significantly different, when editing arena settings…
Solution: A new calibration line that is significantly smaller or bigger than the others may have been accidentally added. This causes a conflict between the existing calibration lines which

measure the arenas at 64.5cm. To solve, delete the new calibration line that was accidentally created.

Problem: RECORD hangs at the reward delivery stage of the trial…
Solution: Sometimes during a trial, the RECORD system will hang at any stage of the trial, especially the reward delivery. This is normal. Reward delivery is heavily dependent on animal detection and this may sometimes cause delays depending on the animal's position. If the system does not respond after 10 - 20 seconds, manually stop the trial and troubleshoot the detection settings or arena settings. *Make sure to close all command windows (small black windows) before starting the next trial.* This will ensure that the RECORD system will be responsive for the next trial.

Firmware troubleshooting

It is rare for the firmware running on the microcontroller to act up. Firmware is debugged before uploading it to the microcontroller. Fixes to the firmware will require C language programming skills, but a good solution that will work most of the time is to turn the microcontroller(s) off, wait 10 seconds, then turn it back on.

Resource table

[Under construction]