# General Arduino Protocol

This document contains a general overview of using Arduino and a brief review of the types of scripts that we include with the system.

A detailed overview of the Arduino language and program structure are beyond the scope of this protocol. For a more detailed overview refer to the Arduino website ([link](https://www.arduino.cc/reference/en/)).

## General Overview

* Arduinos operates independently of the computer connected to it. Once a script is uploaded to the Arduino, the script will run continuously until the Arduino is loses power or is reset.
* Arduinos can be reset through the following methods
  + Opening a serial connection with the Arduino (e.g. opening the serial monitor in the Arduino IDE or opening a serial connection through python).
  + Connecting and disconnecting the USB
  + Reset button on the Arduino

*Note: Make sure you are aware of the current state of the Arduino program. When using behavioral programs, reset the Arduino prior to head-fixing a subject to ensure that the program is in the starting state.*

* Each script we have included with this system is composed of 3 major sections
  + Definitions:
    - Each variable is defined. Crucially, the datatype must be set as the variable is defined and will limit the range of values the variable can represent.
    - Dependencies are included
    - Values for many parameters are set
  + void setup ():
    - code that is ran once from top to bottom
    - Behavioral scripts include a line that waits for a serial command from the computer to begin.
  + void loop ():
    - code that is ran repeatedly from top to bottom

## IDE Overview

Serial Monitor

### Main screen

Graphical user interface, text

Description automatically generated

Upload (Verify script and send to current Arduino)

Verify (check for Arduino syntax)

COM for current Arduino

Arduino Script

Serial Monitor

### Tools menu

Manage libraries to download external libraries

Graphical user interface, application

Description automatically generated

Set board type of current Arduino

Set current Arduino based on COM port

## Scripts and Serial Monitor Overview

* Arduino scripts (suffix = .ino) must be contained within folders of the same name as the file.
* To open an Arduino script, double click on the .ino file.
* To send script to Arduino, click the “Upload” button
  + You cannot send an script to an Arduino with an active serial connection from another instance of the IDE or external program. Make sure to close these out before attempting to send script to the Arduino.
* To interact and/or monitor the Arduino, click the Serial Monitor button
  + This will connect to the Arduino and reset the script currently on the Arduino (edits to the script will not be reflected on the Arduino until the script is uploaded).
* Serial Monitor

Button to send string to Arduino

Graphical user interface, text, application, Word, email

Description automatically generated

Baud rate for serial communication

String to send

to Arduino

Display for serial prints from Arduino

* + Multiple “helper” scripts use the Serial Monitor string input to control the state of the Arduino. To do this, simply type out a string in the box to the left of the “Send” button and click Send to communicate it to the Arduino.
* Check to ensure that the baud rate is set correctly.
  + Baud rate is set within the Arduino program with Serial.begin(*baudrate*)

A picture containing diagram

Description automatically generated

* + For programs included with this system, the default baud rate is set to 115200. Other scripts may use different buad rates.
  + If the baud rate is not set correctly, the strings produced by the Arduino will not be decoded properly and will be printed with jumbled symbols

Graphical user interface, text, application, Word

Description automatically generated

* Behavioral scripts will require that a string is sent to the Arduino before the program begins
  + For testing a program, open the Serial Monitor and click the “Send” button to send an empty string and start the behavioral program.
  + During real behavioral sessions, we will use a python script to open the serial connection, start the Arduino, and record the serial (see Python section below).

## External libraries

* You will need to install the MPR121 library in order to compile and upload scripts that contain lick detection.
  + Installed through Tools > Manage Libraries > search MPR121

Graphical user interface, text, application, email

Description automatically generated

* + Make sure to install with all dependencies

## write\_serial.py Overview

* The *write\_serial.py* script is used to restart the Arduino, send a start command, and then record the Serial prints while the Arduino is running.
* Can be run using any IDE or directly from the command line. We used Sublime Text 3 for the example shown here.
* There are 3 parameters that you will need to set:

Text

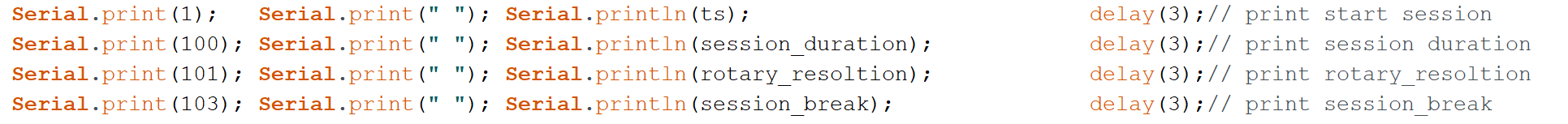
Description automatically generated

* + COMPORT: string identifier for the comport of the Arduino (check the Arduino Tools > Port for Com ID)
  + BAUDRATE: integer baud rate (must match comport set in Arduino script)
  + Subject: string for subject ID (must be unique for each session)
* When executed the program will do the following
  + Generate file name based on current date and subject id:
    - General format: *yyyy\_mm\_dd\_subject*
    - If there is a conflict, a string suffix will be added for the time with the following format: *\_hhmmss*
  + Open serial communication with Arduino based on COMPORT and BAUDRATE
  + Generate and then append a csv file with each line printed by the Arduino each
    - *Note: the program only works with numeric and space prints from the Arduino and will not work with strings of characters*
  + Closes serial communication and stops executing when a “0” is printed in the first position of the print from the Arduino.
    - *Note: If you edit the Arduino scripts, do not include a print with 0 in the first position*

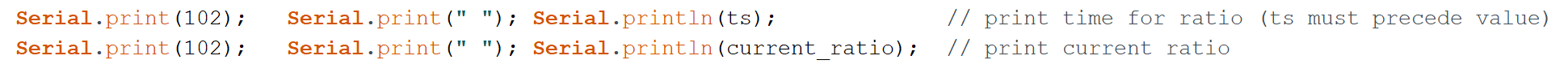
## Raw data structure

All behavioral programs are written to have a uniform data structure consisting of two columns of numerical data that fall into 3 categories

1. Parameters: stable parameters for the session (1 row). These values print once at the start of the session.
   1. First column: parameter ID
   2. Second column: parameter value



1. Dynamic parameters: parameters can change values at different points throughout a session (**2 rows / change in parameter**)
   1. First column first and second rows: parameter ID
   2. Second column first row: time stamp for the onset of the current parameter (ms)
   3. Second column second row: current value of the parameter



1. Events: events occurring during the behavioral session
   1. First column event ID
   2. Second column timestamp (ms)



* All of the IDs for events, parameters, and dynamic parameters are listed in *key\_events.csv*. This table is used to decode data within each raw Arduino csv file.
  + If you add additional events, follow the format outlined here in the Arduino print, and add a corresponding row to the *key\_events.csv* file.

## Trouble shooting quick guide:

* Cannot upload script to Arduino

Text

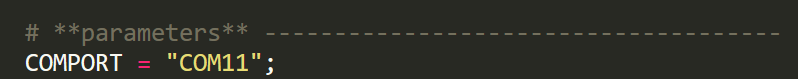
Description automatically generated

* + Check to ensure that all other serial communication with the Arduino is closed. If the problem persists, disconnect / reconnect Arduino USB
* Strange symbols in serial print via Serial Monitor
  + check baud rate in script and serial connection
* Cannot connect to Arduino with *write\_serial.py*

A picture containing graphical user interface

Description automatically generated

* + Check to ensure that all other serial communication with the Arduino is closed. If the problem persists, disconnect / reconnect Arduino USB
  + Check to ensure that COMPORT is set to the correct Arduino



* UnicodeDecodedError with *write\_serial.py*

A picture containing text

Description automatically generated

* + Check to ensure that baud rate in write\_serial.py matches baud rate set in Arduino script
  + Check to ensure that COMPORT is set to the correct Arduino

*Write\_serial.py Arduino script*

Graphical user interface

Description automatically generated A picture containing diagram

Description automatically generated