Micro Fluidic Handling System – Design Doc

Purpose: Provide an easy-to-use, modular, and compact microfluidic platform for PML-type chips (but can be used without them).

Hardware: Compact form-factor microfluidic handling devices that can be easily purchased and developed into a full microfluidic handler. Our system currently contains an ISMATEC-REGLO 3-channel peristaltic pump, A Rheodyne-Titan EX manifold switch, and a lab-made automated collection stage.

Hardware Plans: A microscope compatible collection stage is in the works. We’re considering designing our own pump. (96-well plate feeder?)

Software: Simple graphical user interface built in python and using basic serial communication. Device modules written in python. GUI gives user basic control over pump, manifold switch, and collection stage. Modularity built-in to some of the architecture.

Software Plans: Include modular software architecture expansion. Cleaning up of background architecture so that it’s more modular and patch-suitable (i.e. it can work for a range of pumps and manifold types, etc). Adding a visualizing feature to see the plates and flows (if other sensors installed).

**GUI.py**

|  |  |
| --- | --- |
| **Collection.py** |  |

|  |  |  |
| --- | --- | --- |
| **Modules** | | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | | [fakeSerial](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\fakeSerial.html) | [sys](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\sys.html) | [time](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\time.html) |  | |

|  |  |  |
| --- | --- | --- |
| **Classes** | | |
|  |  | [builtins.object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object)  [Collectador](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\Collection.html#Collectador)   |  |  |  | | --- | --- | --- | | class **Collectador**([builtins.object](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\builtins.html" \l "object)) | | | |  | [Collectador](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\Collection.html#Collectador)(my\_port)   This class is used for control of stepper motors. A serial connection is established with the microcontroller controlling the stepper motors.   Attributes:     baud (int): The baud rate the serial connection is using.     port (string): the microcontrollers port for the serial connection.     ser (serial Object): Instance of serial [object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object) representing the serial connection.     serialConnected (bool): True/False of whether the serial connection was established.     position (int): the position that the XY Stage is currently at.     uniqueID (string): the unique identifier the microcontroller returns for automatic port selection. | | |  | Methods defined here:  **\_\_init\_\_**(self, my\_port)  The constructor for the [Collectador](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\Collection.html" \l "Collectador) class.   Parameters:     my\_port (string): the microcontrollers port for the serial connection.  **chop\_return**(self, ret)  This method modifies the output of the pump which comes with a carriage return and newline at the end of itself. The carriage return and newline is cut off here.   Returns:     Output of the pump without the carraige return and newline at the end.  **eject**(self)  This method sends the command "E" to the microcontroller which is programmed to move the XY Stage to the eject position when receiving this command. Simulatenously also updates the current position of the XY Stage.  **get\_info**(self)  This method sends the command "?" to the microcontroller which is programmed to send back a unique ID. The attribute uniqueID is set to the microcontrollers response.   Returns:     response (string): The unique ID that the microcontroller sends back through the serial connection.  **last\_site**(self)  This method sends the command "L" to the microcontroller which is programmed to move the XY Stage to the last position when receiving this command. Simulatenously also updates the current position of the XY Stage.  **next\_site**(self)  This method sends the command "N" to the microcontroller which is programmed to move the XY Stage to the next position when receiving this command. Simulatenously also updates the current position of the XY Stage.  **reset**(self)  This method sends the command "Z" to the microcontroller which is programmed to move the XY Stage to the origin when receiving this command. Simulatenously also updates the current position of the XY Stage.  **send**(self, cmd)  This method sends a command across the serial connection.   Parameters:     cmd (string): The command or string that is to be sent to the microcontroller.  **serial\_connect**(self)  This method establishes the serial connection with the microcontroller.   Once the comport of the pump is known we open a serial connection to it using pySerial.  Data descriptors defined here:  **\_\_dict\_\_**  dictionary for instance variables (if defined)  **\_\_weakref\_\_**  list of weak references to the object (if defined) | |

|  |  |
| --- | --- |
| **Pump.py** |  |

|  |  |  |
| --- | --- | --- |
| **Modules** | | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | | [serial](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\serial.html) | [sys](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\sys.html) | [time](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\time.html) |  | |

|  |  |  |
| --- | --- | --- |
| **Classes** | | |
|  |  | [builtins.object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object)  [ThreePump](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\Pump.html#ThreePump)   |  |  |  | | --- | --- | --- | | class **ThreePump**([builtins.object](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\builtins.html" \l "object)) | | | |  | [ThreePump](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\Pump.html#ThreePump)(my\_port)   This class is used for control of the Pump. A serial connection is established with the microcontroller controlling the Pump.   Attributes:     baud (int): The baud rate the serial connection is using.     port (string): the microcontrollers port for the serial connection.     ser (serial Object): Instance of serial [object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object) representing the serial connection.     serialConnected (bool): True/False of whether the serial connection was established.     calibrationvolume (int): amount of fluid used for calibration.     calibrationunit (string): the unit for the amount of fluid used for calibration.     isOn (bool): True/False on whether the pump is on or not. | | |  | Methods defined here:  **FormatVolume**(self, V, unit)  This method formats the volume.   Parameters:     V (int): The amount of volume.     unit (string): The unit the volume should be set in.  **\_\_init\_\_**(self, my\_port)  The constructor for the TwoSwitch class.   Parameters:     my\_port (string): the microcontrollers port for the serial connection.  **abort\_calibration**(self, channel)  This method sends a command to the microcontroller which is programmed to abort calibration of the pump only for the channel chosen.   Parameters:     channel (int): the channel that will abort calibration.  **calibrate**(self, channel)  This method sends a command to the microcontroller which is programmed to calibrate the pump only for the channel chosen.   Parameters:     channel (int): the channel that will be calibrated.  **chop\_return**(self, ret)  This method modifies the output of the two switch which comes with a carriage return and newline at the end of itself. The carriage return and newline is cut off here.   Returns:     Output of the pump without the carraige return and newline at the end.  **send**(self, cmd)  This method sends a command across the serial connection.   Parameters:     cmd (string): The command or string that is to be sent to the microcontroller.  **send\_return**(self, cmd)  This method uses the serial connection opened instance of pump and sends the text written in cmd across that connection.   Returns:     response (string): The response or what the microcontroller returns.  **serial\_connect**(self)  This method establishes the serial connection with the microcontroller.   Once the comport of the pump is known we open a serial connection to it using pySerial.  **setDefaults**(self)  This method sets the deafults for the pump.  **setDir**(self, channel, direction)  This method sets the direction for the flow of fluid.   Parameters:     channel (int): The channel flow rate direction will be set.     direction (string): The direction for the pump CW or CCW.  **setFlow**(self, channel, flowrate)  This method sets the flow rate of the pump.   Parameters:     channel (int): The channel whose flow rate will be set.     flowrate (int): The flow rate which is to be set.  **setMeasuredVolume**(self, channel, volume)  This method sets the measured volume.  **setTargetCalibrationVolume**(self)  This method sets the target calibration volume.  **start**(self, channel)  This method sends a command to the microcontroller which is programmed to start the pump only for the channel chosen.   Parameters:     channel (int): the channel that will start.  **start\_all**(self)  This method sends a command to the microcontroller which is programmed to start the pump for all channels.  **stop**(self, channel)  This method sends a command to the microcontroller which is programmed to stop the pump only for the channel chosen.   Parameters:     channel (int): the channel that will stop.  **stop\_all**(self)  This method sends a command to the microcontroller which is programmed to stop the pump for all the channels.  Data descriptors defined here:  **\_\_dict\_\_**  dictionary for instance variables (if defined)  **\_\_weakref\_\_**  list of weak references to the object (if defined) | |

|  |  |  |
| --- | --- | --- |
| **Functions** | | |
|  |  | **get\_ports**()  This method uses serial module's comports command to get and return serial ports. |

|  |  |
| --- | --- |
| **Mswitch.py** |  |

|  |  |  |
| --- | --- | --- |
| **Modules** | | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | | [serial](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\serial.html) | [sys](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\sys.html) | [time](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\time.html) |  | |

|  |  |  |
| --- | --- | --- |
| **Classes** | | |
|  |  | [builtins.object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object)  [MSwitch](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\Mswitch.html#MSwitch)   |  |  |  | | --- | --- | --- | | class **MSwitch**([builtins.object](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\builtins.html" \l "object)) | | | |  | [MSwitch](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\Mswitch.html#MSwitch)(my\_port)   This class is used for control of the Manifold Switch. A serial connection is established with the microcontroller controlling the Manifold Switch.   Attributes:     baud (int): The baud rate the serial connection is using.     port (string): the microcontrollers port for the serial connection.     ser (serial Object): Instance of serial [object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object) representing the serial connection.     serialConnected (bool): True/False of whether the serial connection was established.     res (int): The reservoir chosen.     uniqueID (string): the unique identifier the microcontroller returns for automatic port selection. | | |  | Methods defined here:  **\_\_init\_\_**(self, my\_port)  The constructor for the [MSwitch](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\Mswitch.html" \l "MSwitch) class.   Parameters:     my\_port (string): the microcontrollers port for the serial connection.  **chop\_return**(self, ret)  This method modifies the output of the pump which comes with a carriage return and newline at the end of itself. The carriage return and newline is cut off here.   Returns:     Output of the pump without the carraige return and newline at the end.  **get\_info**(self)  This method sends the command "?" to the microcontroller which is programmed to send back a unique ID. The attribute uniqueID is set to the microcontrollers response.   Returns:     response (string): The unique ID that the microcontroller sends back through the serial connection.  **send**(self, cmd)  This method sends a command across the serial connection.   Parameters:     cmd (string): The command or string that is to be sent to the microcontroller.  **serial\_connect**(self)  This method establishes the serial connection with the microcontroller.   Once the comport of the pump is known we open a serial connection to it using pySerial.  **set\_reservoir**(self, res)  This method sets the reservoir to pull fluid from.  Data descriptors defined here:  **\_\_dict\_\_**  dictionary for instance variables (if defined)  **\_\_weakref\_\_**  list of weak references to the object (if defined) | |

|  |  |
| --- | --- |
| **TwoSwitch.py** |  |

|  |  |  |
| --- | --- | --- |
| **Modules** | | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | | [fakeSerial](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\fakeSerial.html) | [sys](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\sys.html) | [time](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\time.html) |  | |

|  |  |  |
| --- | --- | --- |
| **Classes** | | |
|  |  | [builtins.object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object)  [TwoSwitch](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\TwoSwitch.html#TwoSwitch)   |  |  |  | | --- | --- | --- | | class **TwoSwitch**([builtins.object](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\builtins.html" \l "object)) | | | |  | [TwoSwitch](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\TwoSwitch.html#TwoSwitch)(my\_port)   This class is used for control of the two switches. A serial connection is established with the microcontroller controlling the two switches.   Attributes:     baud (int): The baud rate the serial connection is using.     port (string): the microcontrollers port for the serial connection.     ser (serial Object): Instance of serial [object](file:///C:\Users\Adiel%20Hernandez\Google%20Drive\Developer\Senior%20Design\fluidic_handler\builtins.html#object) representing the serial connection.     serialConnected (bool): True/False of whether the serial connection was established.     uniqueID (string): the unique identifier the microcontroller returns for automatic port selection.     willRecirculate (bool): True/False as to if the two switches are set to recirculate. | | |  | Methods defined here:  **\_\_init\_\_**(self, my\_port)  The constructor for the [TwoSwitch](file:///C:\\Users\\Adiel%20Hernandez\\Google%20Drive\\Developer\\Senior%20Design\\fluidic_handler\\TwoSwitch.html" \l "TwoSwitch) class.   Parameters:     my\_port (string): the microcontrollers port for the serial connection.  **chop\_return**(self, ret)  This method modifies the output of the two switch which comes with a carriage return and newline at the end of itself. The carriage return and newline is cut off here.   Returns:     Output of the pump without the carraige return and newline at the end.  **getIdentifier**(self)  This method sends the command "?" to the microcontroller which is programmed to send back a unique ID. The attribute uniqueID is set to the microcontrollers response.   Returns:     response (string): The unique ID that the microcontroller sends back through the serial connection.  **send**(self, cmd)  This method sends a command across the serial connection.   Parameters:     cmd (string): The command or string that is to be sent to the microcontroller.  **serial\_connect**(self)  This method establishes the serial connection with the microcontroller.   Once the comport of the pump is known we open a serial connection to it using pySerial.  **setCollect**(self, channel)  This method sends a command to the microcontroller based on what channel needs to be set to recirculate.   Parameters:     channel (int): The fluidic channel chosen to be set to collect.  **setRecirculate**(self, channel)  This method sends a command to the microcontroller based on what channel needs to be set to recirculate.   Parameters:     channel (int): The fluidic channel chosen to be set to recirculate.  Data descriptors defined here:  **\_\_dict\_\_**  dictionary for instance variables (if defined)  **\_\_weakref\_\_**  list of weak references to the object (if defined) | |

**Arduino**