PHiL Users Guide

PHiL is a general purpose, multi-signal electronics test system. It enables you to drive peripheral electronics and/or read what is being driven. It includes a board with over 30+ signal connectors on pin headers, a user interface providing control of all of the signals and a drag 'n drop dashboard creator for a customized user interface.

This guide will show you how to use PHiL.

Installation - The User Interface

Install PHiL

Windows 10

The USB stick you received has a setup.exe for installation. Follow through the process and let it restart your machine.   A Phil Shortcut will be placed on your desktop.

Mac and Linux

Parlay is the multi-purpose application to operate Phil on Mac and Linux at this time. There is not an installer app. however it is a Python package that can be installed from the command line with pip (using Python 2.7):

```
sudo pip install parlay
```

MacOS: Do not use the pre-installed Python for Mac. Instead install python 2.7 from Python.org.: https://www.python.org/downloads/release/python-2715/.   After installation, run at the command prompt: sudo pip install parlay

Run PHiL

Windows Installer Application

If you used the Windows installer, you are now ready to get started.

1. Connect the PHiL board to your computer

2. run Phil from your desktop pressing "P" icon on your desktop for Windows, If you get a Windows defender Firewall error you can select either private networks or public networks but you will need to allow access.

Phil will attempt to auto-connect to the PHiL board and discover what is available. When discovery is complete, you should receive notification that 82 items were discovered.

You are now ready to start using PHiL.

PIP Installed Parlay Application

If you installed Parlay via pip, open a command terminal and type:

```
parlay
```

This will automatically bring up the browser application.   We suggest using Chrome as your default browser. Some features may not be supported in other browsers.

You will need to manually connect the PHiL board Protocol.

- [ ] Press "Protocols" on the left panel
- [ ] select "open protocol" on the bottom right of the dialog.
- [ ] Select PCOMSerial in the dropdown menu.
- [ ] Select the port matching your connected PHiL board USB port.
- [ ] Press "Discovery" for the application to discover what is on the board. After it completes, you should an indicator that 82 items were discovered.

You are now ready to start using Parlay - the user interface for the PHiL board.

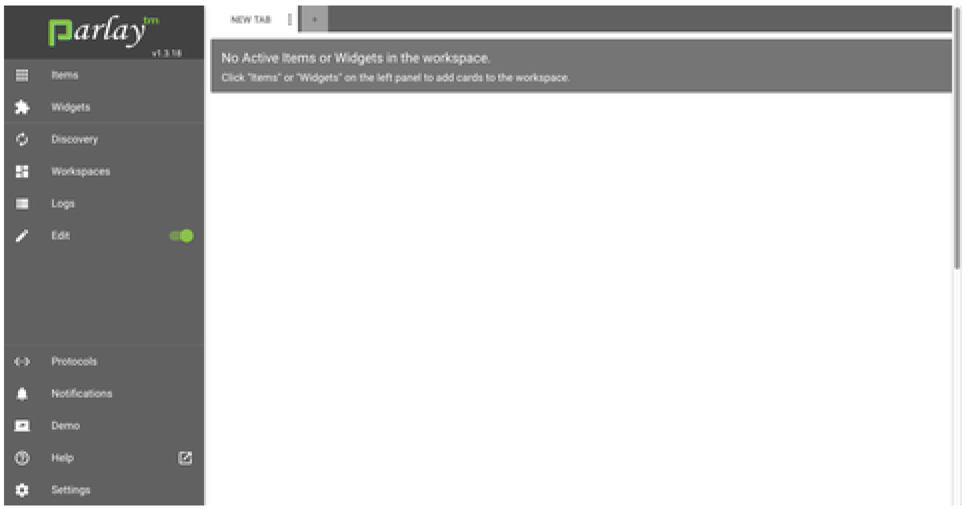
Navigating Phil - a Tutorial

The Phil application is your "command central". It lets you control individual pin items and create a custom dashboard with widgets controlling your signals.

Connect the PHiL board

- [x] First, **connect your PHiL board to your computer** via the USB cable provided. You should see an LED light blinking at 1hz.
- [x] Execute Parlay. On startup, Parlay will attempt to connect to your board, discover the items available

Your initial screen will look like the following:



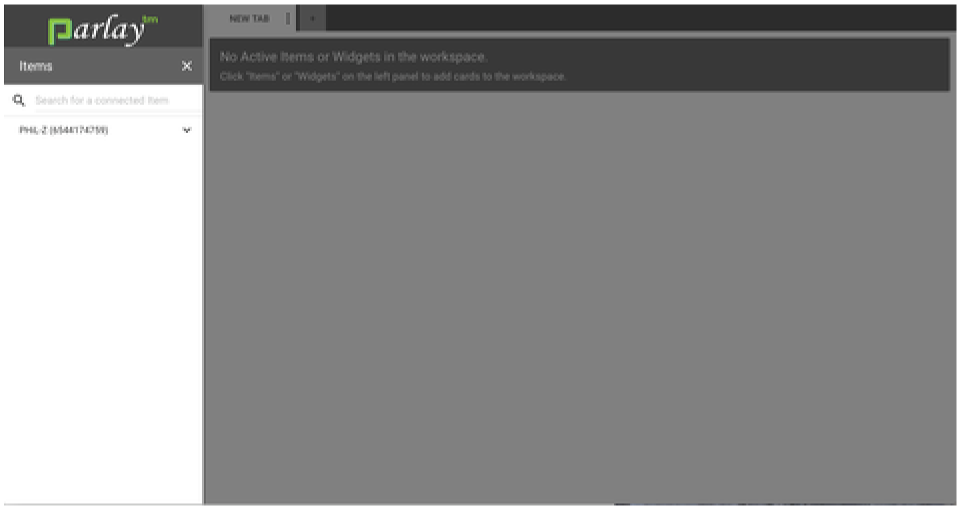
No Active Items or Widgets in the workspace.
Click "Items" or "Widgets" on the left panel to add cards to the workspace.

This is an empty workspace. When you start, your last workspace will come up by default.

Initial Navigation

The first step is to view the "Items" available on the board.

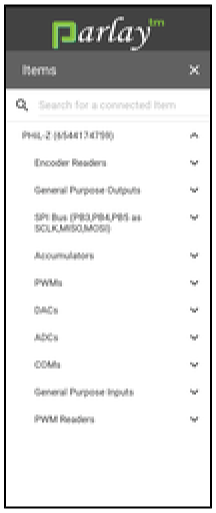
- ✓ Press "Items" on the left panel. You should see something like the following on the left panel:



No Active Items or Widgets in the workspace.
Click "Items" or "Widgets" on the left panel to add cards to the workspace.

This shows this is the PHiL-Z board. In parenthesis is the serial number of your board.

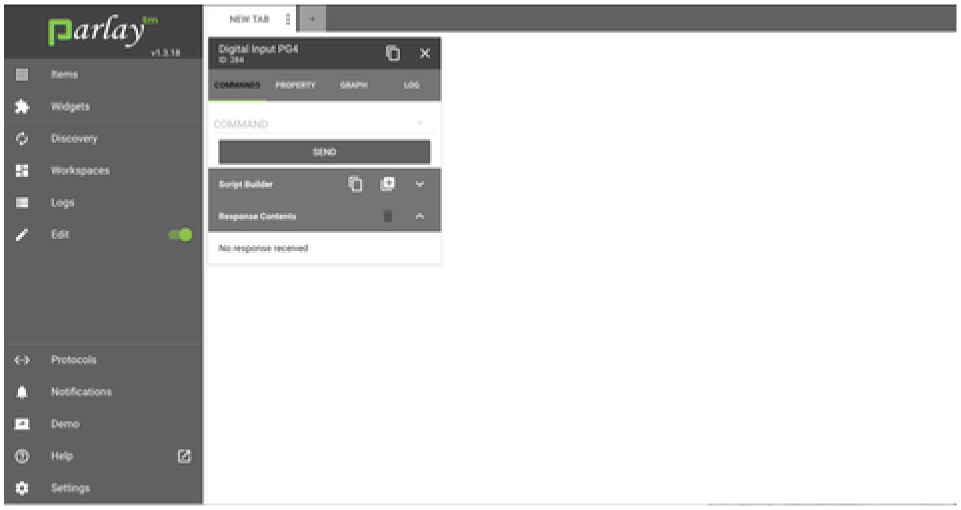
- ✓ Press the down arrow next to the PHiL-Z header, and item categories are seen:



These are the categories of signals controllable/Viewable on the PHiL board.

- ✓ Expand "General Purpose Inputs" and you will see a list of items available as GPIO inputs.
- ✓ Mouse Click on "Digital Input PG4"

You will see an ***Item Card*** get placed in the workspace. Click on the work space to move focus from the *Item* panel to the workspace.



Each *Item* represents a controllable signal.   The pin label (as seen on the PHiL board) is part of the item name. For example: Digital Input PE11 corresponds to the PE11 label on your PHiL board.

Item Cards

Lets explore the Item Card. In this card you can:

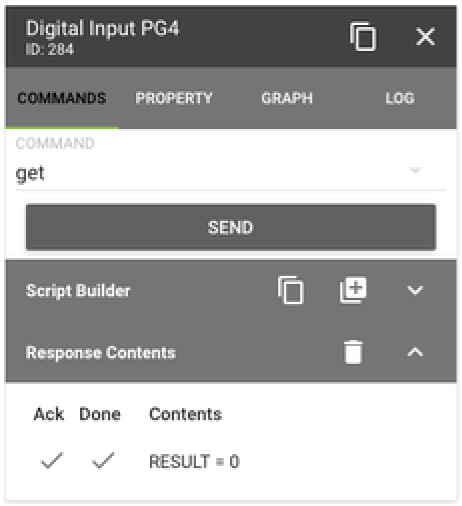
- Send Commands to the Item, and see the response. Each category of item will have different commands available that are appropriate for it.
- View and change Properties to the Item. Properties can be data read from the pin, data set to the pin, or configuration parameters of the item.
- Graph Properties of the Item. If the property is data read, such as the value of an ADC, you can graph it.
- Review the log of Commands going to and from the Item**.**
- View or copy to your clipboard the python script that corresponds to Commands chosen for use in a Python script. Or create a widget that will execute the Command chosen.

Lets start with Commands.

- ✓ Press on the greyed out "COMMAND" to show the list of commands.

✓ Select "get", and press the "SEND" button. This reads the value of the GPIO input on pin PG 4.

You will see that the Response Contents get filled in as shown below:

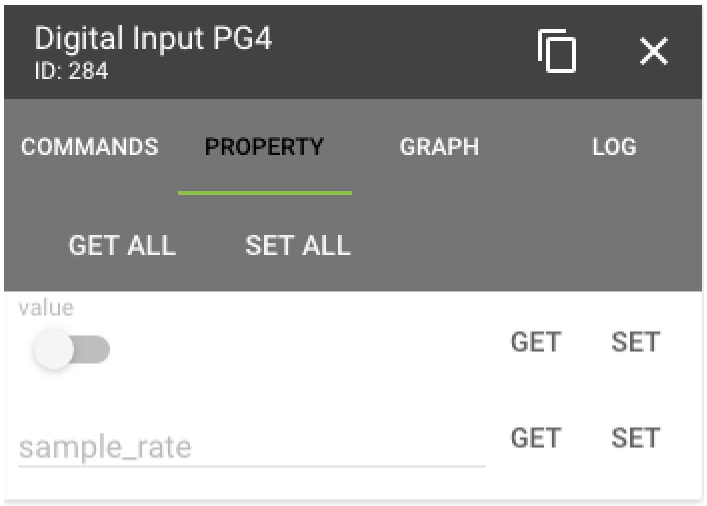


Ack	Done	Contents
✓	✓	RESULT = 0

The "Ack" check means the messages was properly received by the board. The "Done" check means the board sent a successful response to the command. The "RESULT" field is the data sent back from the board. In this case, the board sent back a 0, meaning the pin is 0 (low). Given that the input is not connected to anything right now, and is floating, it is possible you will see a "1".

Lets look at the properties of this item:

✓ Select the Property tab. You will see the following:



Value and sample_rate are the properties of this item. The values seen are not actual values until you "get" them either using "GET ALL" or "GET"

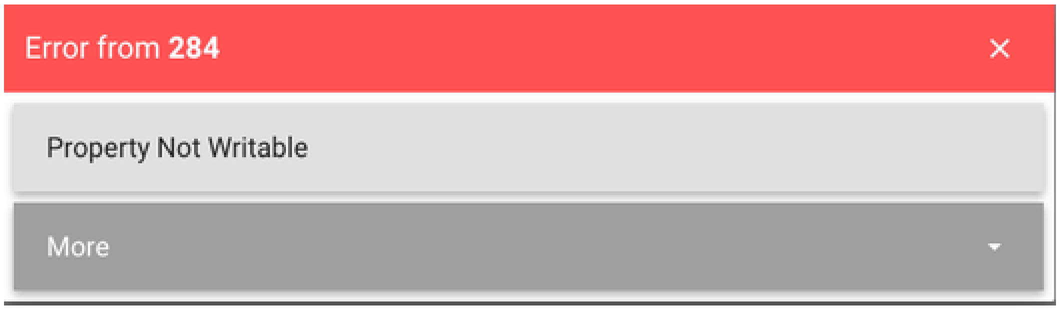
✓ Press GET ALL

You should see that the value is off (0), and the sample_rate is 5. The value is the pin value, which will be either 0 or 1. The sample rate is the rate (in Hz) at which the pin is being sampled and updated by the property variable. Here the property reading the pin 5 times per second.

Just for kicks, lets try to set the value.

✓ Move the toggle to the right, and press "SET"

You will see a big red pop-up declaring an error.


Error from **284**

Property Not Writable

More

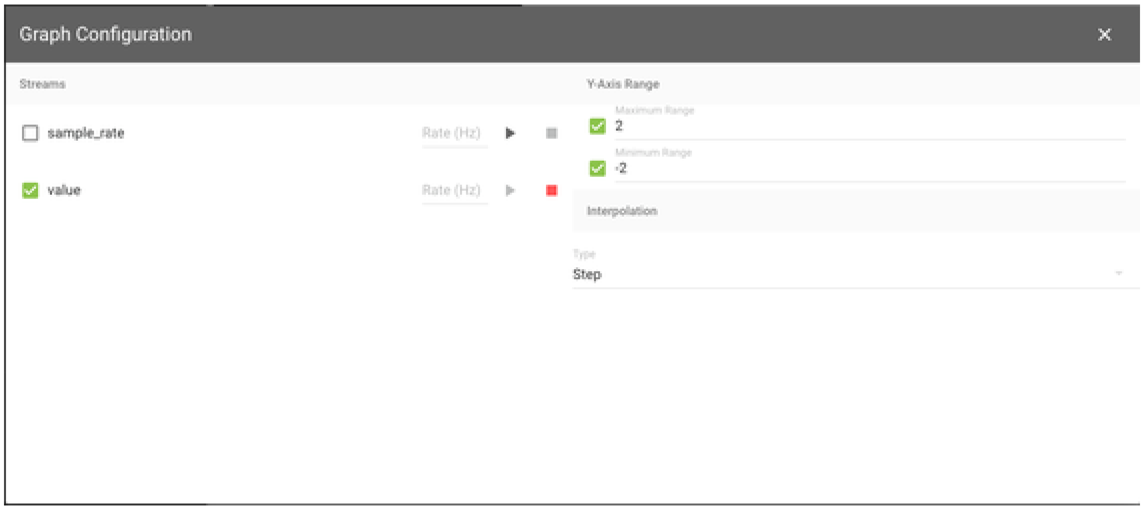
The board sent back an error response to the "SET". This make sense because for an input pin, you can only read the value. You cannot set it.

- ✓ Close the error box.

Lets graph the value of this pin.

- ✓ Select the Graph tab.
- ✓ Check the "value" checkbox
- ✓ Check the minimum and maximum range, and enter 2 for the maximum, and -2 for the minimum.
- ✓ Select the type of graph as "Step"

You should see the following:


Graph Configuration

Streams

- ☐ sample_rate
- ✓ value

Y-Axis Range

- ✓ Maximum Range: 2
- ✓ Minimum Range: -2

Interpolation

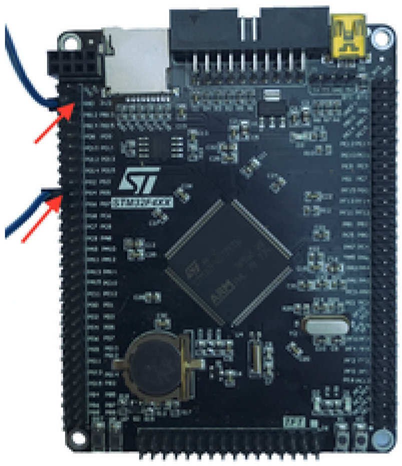
Type: Step

Now close the Graph Configuration with the X at the upper right. You will see the graph of the pin at 0.0.

Setting up a Pin

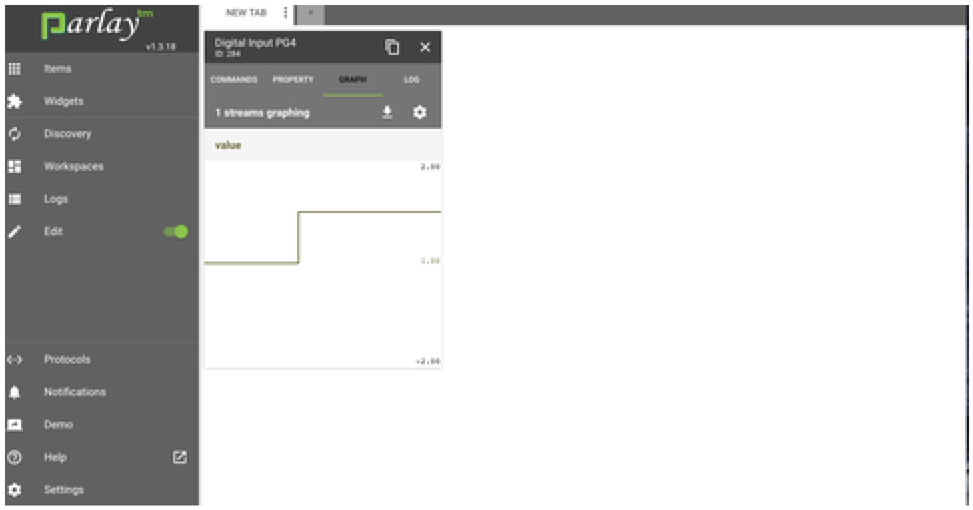
Using the jumper wires provided:

- ✓ carefully connect pin PG4 (GPIO input) to 3v3 as shown below:



For a chart of the pins see: PHiL Pins

When you connect the pin to 3.3v, you will see the graph change as expected.



Now, if you go back to the command "Get" or the Properties and get the "value" property, you will see a "1" return.

Instead of connecting the pin to 3v3, lets choose a GPIO output to connect to.

- ✓ Press on the "Items" on the left panel
- ✓ Navigate to General Purpose Outputs, and select any output pin. In this example, we will choose Digital Output PE13
- ✓ Send the Command "reset" to the output. This will set the pin output to 0. Note there is no "result" after sending this command, because there is no data sent back.
- ✓ Connect your PG4 to PE13 (or the output you selected).
- ✓ Send a "set" command

Again you will see the graph change from 0 to 1.

- ✓ Select the properties tab on the Output Item.

- [✓] Change the value property and press "SET"

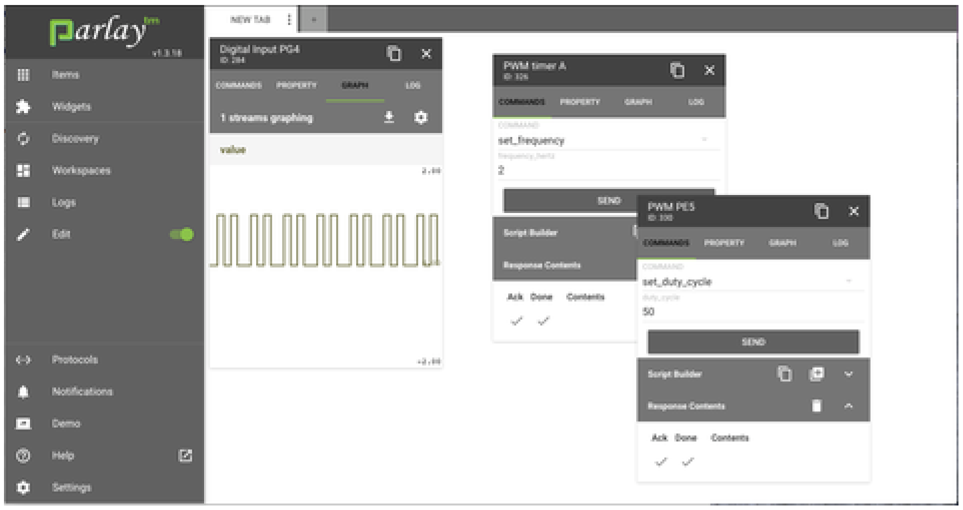
Note that you can change an output either by sending command or by setting its property.

PWM output

Now we will output a PWM to our PG4 input.

- [✓] On the left panel, select Items, and navigate the PWMs, and select PWM timer A (note: actually click on the timer, not the arrow to expand it). The timer card will appear in the workspace.
- [✓] Select the command "set_frequency", and put in a 1 for the frequency, and SEND. This sets our PWM to 1hz (very slow).
- [✓] Now expand the Timer A (press on the arrow) and select PWM PE5. The PWM PE5 item card will be on the screen.
- [✓] Connect your GPIO Input PG4 to PWM output pin PE5 located near the blinking LED (3rd pin up from the bottom)
- [✓] Select the command "set_duty_cycle", setting the duty_cycle to 50. Press SEND.
- [✓] Select command "enable" and press SEND.

You will see the graph of PG4 change with the frequency.



COMMAND set_frequency
frequency_hertz 2

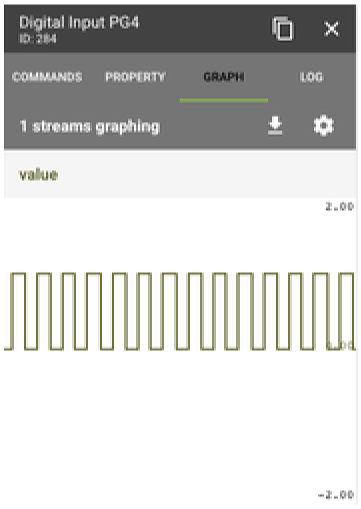
COMMAND set_duty_cycle
duty_cycle 50

Now change the frequency of timer A to be 5 hertz

- [ ] On the Timer A item card, select the set_frequency command, with a frequency_hertz of 5.

You will see the graph go flat - either high or low. This is because our default sample rate is the same as the frequency of the signal. We need to increase the sample rate of the input.

- [✓] Select the Property tab of Digital Input PG4
- [✓] Change the sample_rate to 50 and press SET.
- [✓] Return to the Graph Tab. Now you will see a nice even signal



You may be wondering why you set the PWM *frequency* on the Timer Item, and the *Duty Cycle* on the PWM PE5 Item. This is because the timer on this micro processor can control up to 4 PWMs, with the caveat that they are all on the same shared frequency. However each PWM but can have unique duty cycle. The hierarchy of the Items represents this sharing - 2 PWM's are sharing Timer A.

Read a PWM

With PHiL you can read the duty cycle and frequency of a PWM.

- ✓ Close the PG4 Item card
- ✓ On the Items in the Left Panel navigate to PWM Readers and select "PWM Input PA1"
- ✓ Remove the jumper from PG4 and connect to PA1 (right side in the middle)
- ✓ Select the graph tab and select to graph frequency. You can let the minimum and maximum default to be dynamically chose.
- ✓ On the PWM timer A item, change the frequency to 20000.

Notice the graph of PA1 change to close to 20000. The accuracy of the duty cycle reading will vary, normally being within .1% in the normal PWM ranges, but higher in very low or high ranges. The PWM Input card we chose indicates that it is the best one for lower frequencies (<100kh). The other input pins indicate they can be used for frequencies greater than 5Khz.

Now lets graph the duty cycle.   Configure the graph by pressing the gear box. Select duty_cycle instead of frequency. and close the graph configuration. Now when we change the duty cycle on the PWM PE5 item card, we see the graph follow.

Widgets

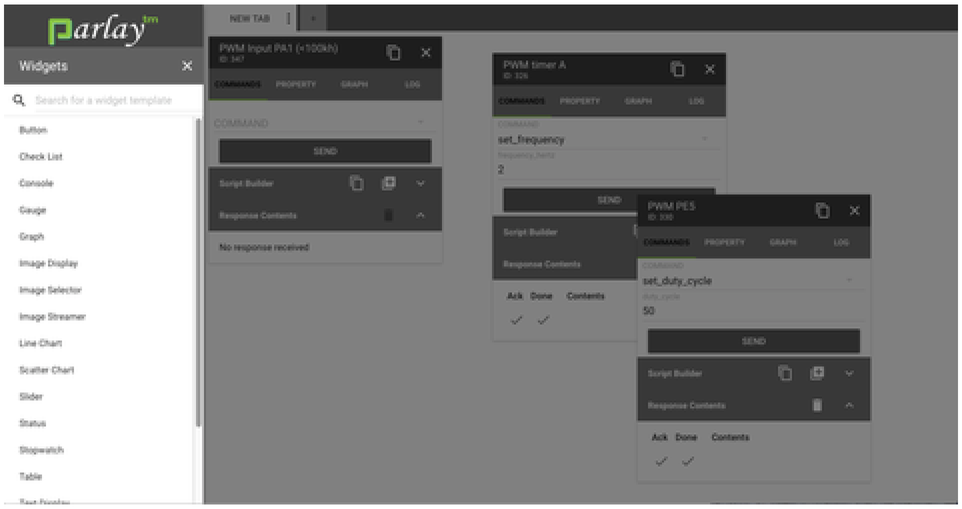
Widgets make for more intuitive control of your items. But they need to be configured.

- ✓ Pressing the widget menu (3 verticle dots), select "edit".

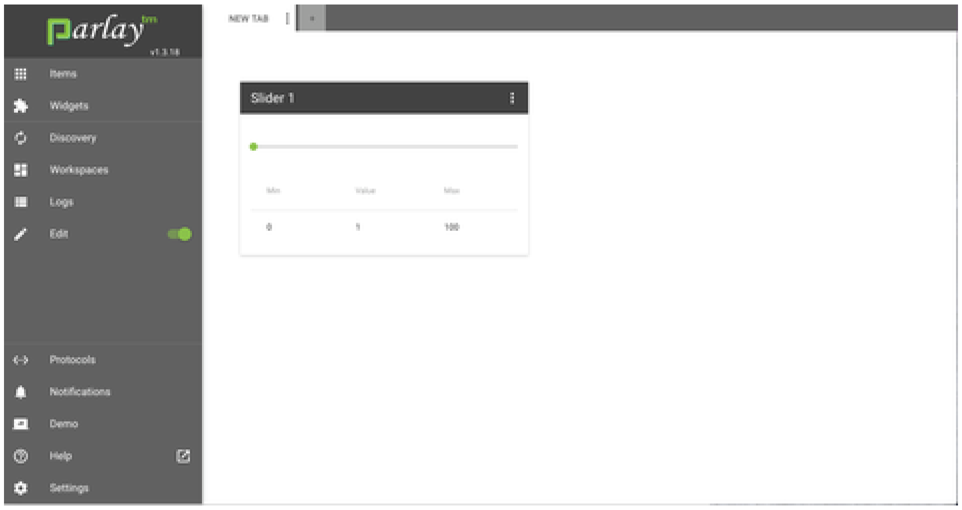
You will see the below:

These can be selected here, but there is an easier way to setup a widget from the item card.

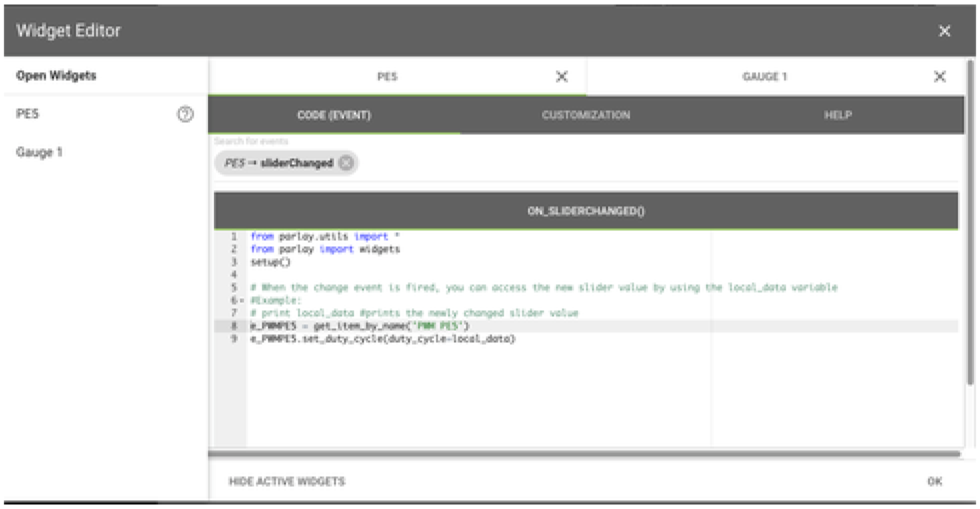
- ✓ On the PWM PE5 card, select the "+" box on the Script Builder. This will allow us to select a widget with the Python code in it to select the duty cycle.



- ✓ Select the Slider Widget
- ✓ Close the item cards on the work space, leaving only the slider widget.



You can see some Python code already written. It will set the duty cycle to 50 when we change the slider.   That is not what we really want. We want the slider to change the duty cycle to whereever we place it.



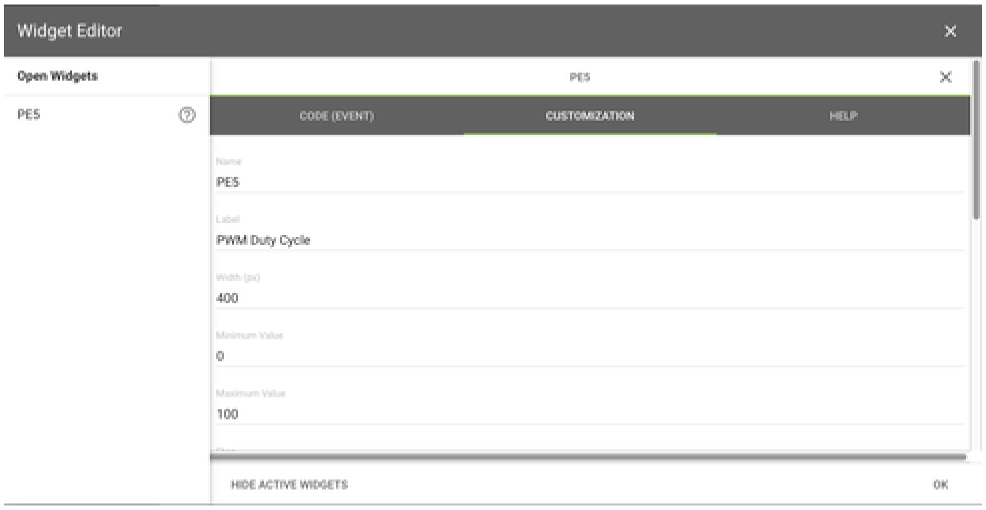
```
from parloy.utils import *
from parloy import widgets
setup()

# When the change event is fired, you can access the new slider value by using the local_data variable
#Example:
# print local_data #prints the newly changed slider value
e_PWMPE5 = get_item_by_name('PWM PE5')
e_PWMPE5.set_duty_cycle(duty_cycle=local_data)
```

- ✓ Change the code from e_PWMPE5.set_duty_cycle(duty_cycle="50") to e_PWMPE5.set_duty_cycle(duty_cycle=local_data).

This will cause any slide of the slider to send a command to PE5 to set the duty cycle to whatever the slider is set to. Now lets customize our slider:

- ✓ Select the Customization tab.
- ✓ Change the name to PE5
- ✓ Put the label as "PWM Duty Cycle"



Name
PE5

Label
PWM Duty Cycle

Width (px)
400

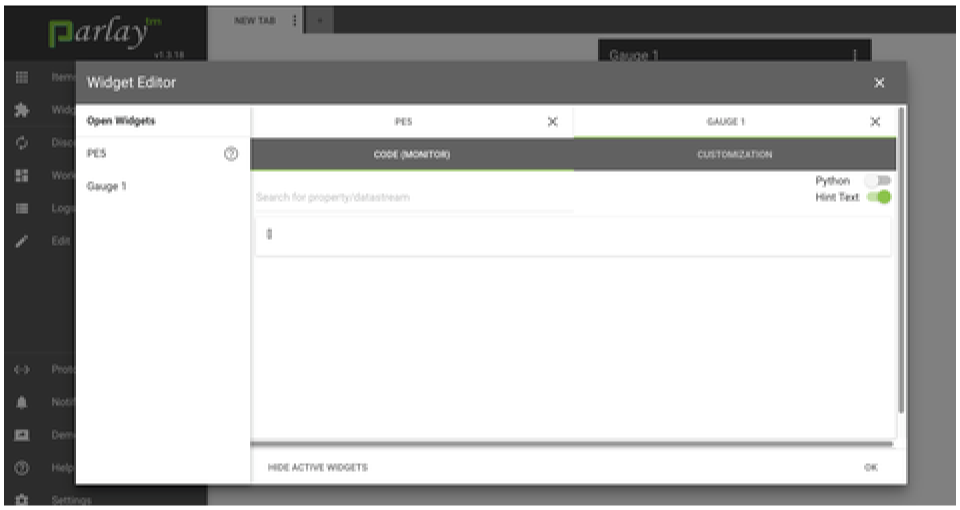
Minimum Value
0

Maximum Value
100

Note that the default range of the slider is 0-100 which works well for a duty cycle in percentage. Other outputs may want to change the range.

Now we will create a gauge widget to show us the duty cycle read.

- ✓ Press OK to close the slider widget editor.
- ✓ On the left Panel select Widget->gauge
- ✓ Edit the gauge widget



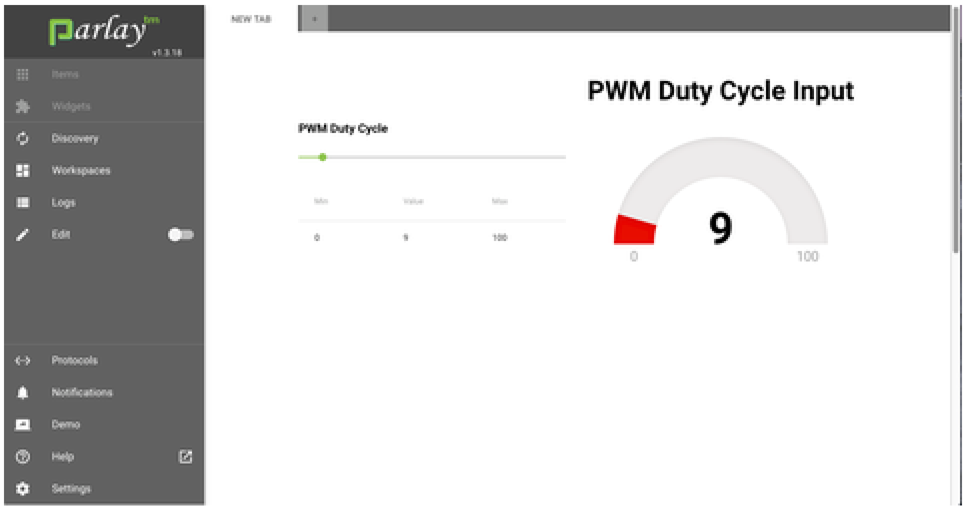
In light text, you will see "Search for property/datastream. This lets us select which property we are monitoring with this gauge.

- ✓ Mouse click on "Search for property/datastream". Scroll down the list until you reach "PWM Input PA1 (<100kh) duty_cycle", and select this input to monitor. This is where your pin is connected to read PWM.
- ✓ Select the Customization tab
- ✓ Name the Widget PA1 and give it a label of "PWM Duty Cycle Input"
- ✓ press OK to close the editor.

Now as you move the slider to change the duty cycle output on pin PE 5, you will see the PA1 gauge change accordingly.

Edit Mode

On the left panel is an Edit toggle. You are currently in edit mode. If you switch that toggle, you will see the following:

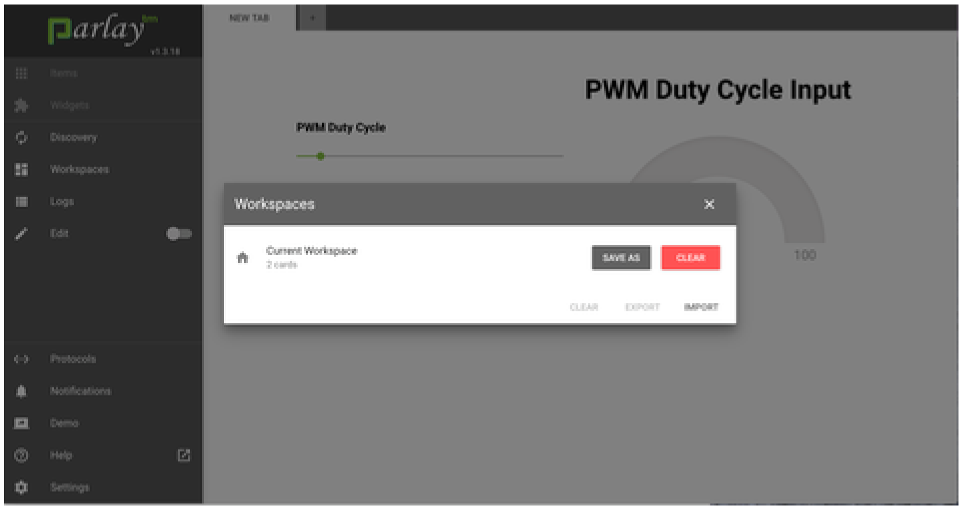


With editing off the names of the items are removed, and only the widget and its label is viewable. The widget cannot be moved or edited. This makes for a cleaner dashboard screen. Note that you cannot select items or widgets when Editing is disabled.

Saving the Workspace

Lets save off this workspace.

- ✓ On the left panel, click on Workspaces. You will see the following:



- ✓ Select "SAVE AS", and give it a name.
- ✓ Save the workspace

Now if you want to go back to that workspace, you can by loading it. You can also export it to a file for someone else to import and use.

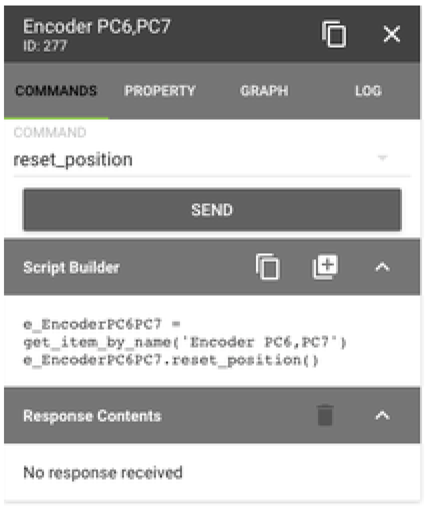
Python Code

While not always necessary, the widgets allow you to write Python code to execute. For example, a button could be programmed to do several things to your items. In this section we will explore some simple scripts to control multiple items. Lets take the hypothetical case of needing to drive a motor to a position, then reading a sensor. We need the following 3 items:

1. Motor PWM
2. Encoder Reader
3. ADC Sensor reader

Without the hardware, we will not try to run this, but we will write some code to respresent what we plan to do.

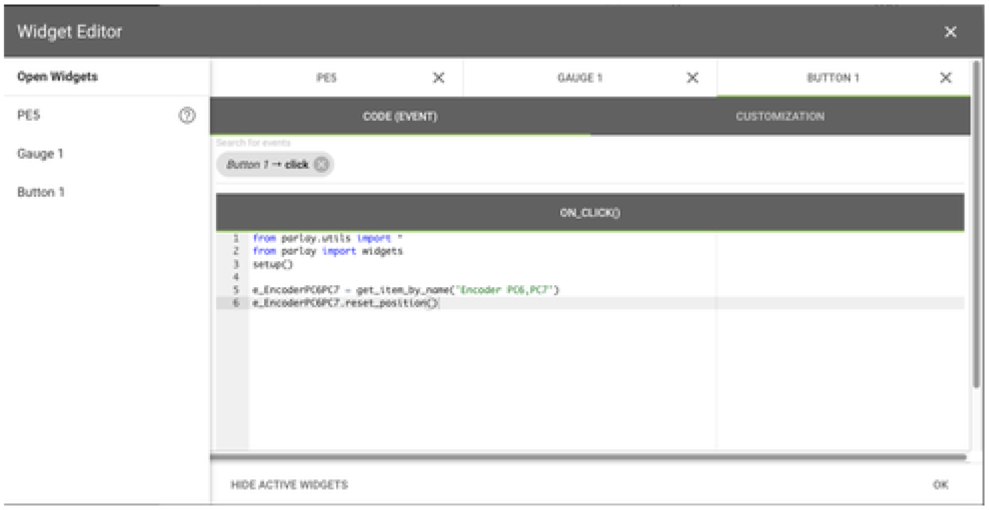
- ✓ Turn on edit mode (if it is still off)
- ✓ From the Items list, select an encoder reader - Encoder PC6, PC7
- ✓ Select the command to reset_position (but no need to press SEND).
- ✓ Expand the Script Builder and you will see the Python code that will reset the position of the encoder (making the currrent position be 0).



```
e_EncoderPC6PC7 =
get_item_by_name('Encoder PC6,PC7')
e_EncoderPC6PC7.reset_position()
```

No response received

- ✓ Click on the + icon and select a button widget. Edit the widget. You will see the code written in its place.

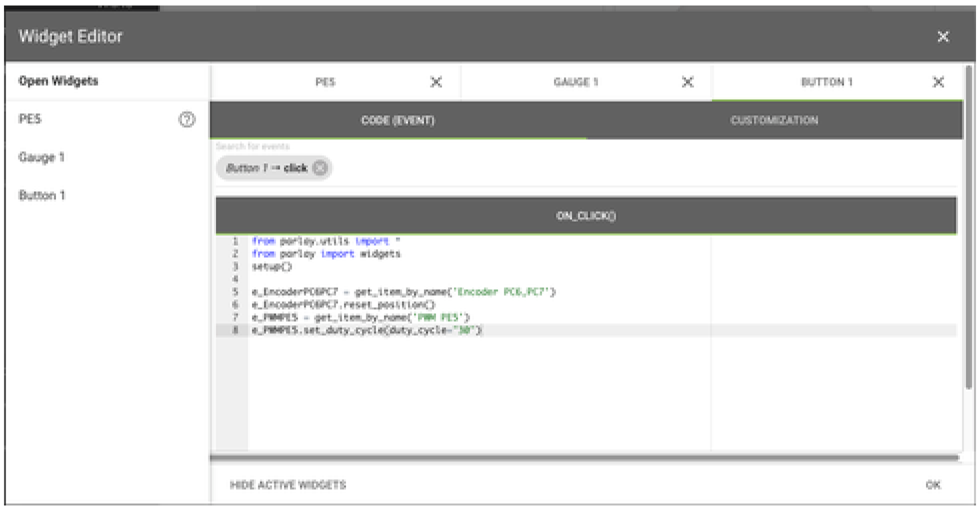


```
from parlay.utils import *
from parlay import widgets
setup()

e_EncoderPC6PC7 = get_item_by_name("Encoder PC6,PC7")
e_EncoderPC6PC7.reset_position()
```

Lets look at the code. We are getting the item object using the get_item_by_name function. Then we can call the items functions or get/set its properties - the same as on the item card. We can either type Python code or copy and paste into the editor code from our items. Lets copy and paste.

- ✓ Close the button and open the item card for PWM PE5 - our PWM we used earlier. Select the command to set_duty_cycle - give it 30 for 30%
- ✓ Press on the clipboard icon next to Script builder, and the commands for setting the duty cycle will be in our clipboard
- ✓ Edit your button again. Right click on your mouse and paste the clipboard contents to your screen. You should see:

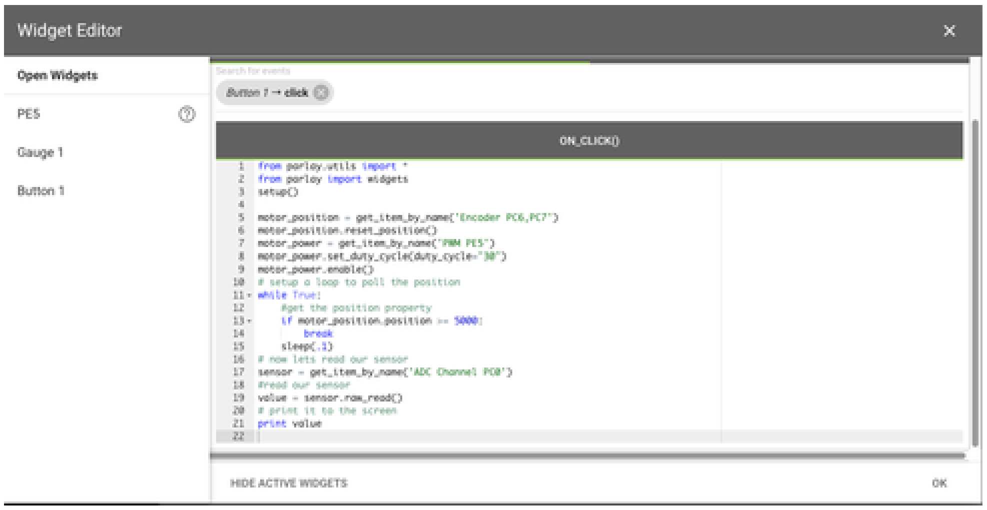


```
from porloy.utils import *
from porloy import widgets
setup()

e_EncoderPC6PC7 = get_item_by_name('Encoder PC6,PC7')
e_EncoderPC6PC7.reset_position()
e_PWMPE5 = get_item_by_name('PWM PE5')
e_PWMPE5.set_duty_cycle(duty_cycle="30")
```

We still need to enable the PWM for it to start, and we want to monitor the encoder until it reaches our goal (say 5000 encoder ticks).. For clarity, we can change the names of our objects to make clear what we are doing. Below is some example code to handle this. We renamed objects to make the code clearer to read. We will poll the position of the motor every .1 second.

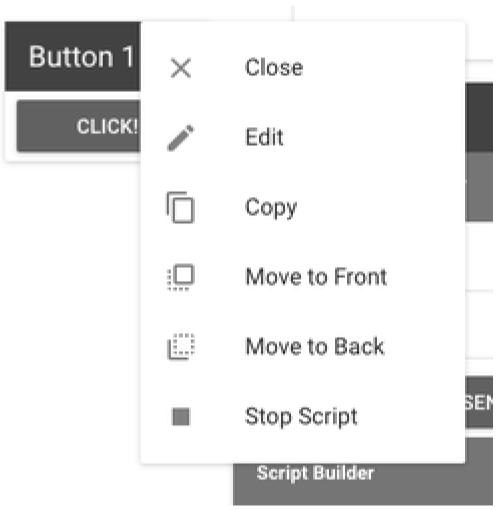
Below is the sample script:



```
from porloy.utils import *
from porloy import widgets
setup()

motor_position = get_item_by_name('Encoder PC6,PC7')
motor_position.reset_position()
motor_power = get_item_by_name('PWM PE5')
motor_power.set_duty_cycle(duty_cycle="30")
motor_power.enable()
# setup a loop to poll the position
while True:
    #get the position property
    if motor_position.position >= 5000:
        break
    sleep(.1)
# now lets read our sensor
sensor = get_item_by_name('ADC Channel PC0')
#read our sensor
value = sensor.raw_read()
# print it to the screen
print value
```

Note that if we run this, we will be in the loop forever because we do not have a motor with encoder attached. However, should you run it, you can stop it with the button's menu Stop Script, as shown below:


Button 1
CLICK!
Close
Edit
Copy
Move to Front
Move to Back
Stop Script
SEN
Script Builder

Using Item Cards

Item cards let you execute commands and read properties of any specific pin signal, such as a GPIO output or PWM output.

Commands

Commands are the functions you can perform on the item. Every item type will have commands appropriate for that item. For example, a general purpose output (GPIO Output) lets you set (3.3v), or reset (0v) the pin. Select the appropriate command from the drop down list when you select ***Commands***.

Using Commands

The available commands for a given item are shown with a drop-down menu. If that command selected has parameters that are necessary to send, additional lines will show below the command with a name of each parameter to send. Type the value for the parameter. For example, a PWM channel command to set the duty cycle will allow you to set the desired duty cycle.

Properties

Properties are the viewable and controllable data/properties of the item. For example, a PWM reader will let you get the current duty cycle, or set a new duty cycle for a PWM in the properties list tab.

Some properties are read only and you will get an error response if you try to set these. These are the properties that represent the signal read from hardware. For example, an input GPIO will show the value read but not allow you to set the value.

Input items, such as PWM, GPIO input, or readers have a sample rate property. This is the rate at which the peripheral is internally read/sampled.

Graphs

Properties can be displayed graphically. This is mostly useful for ADC values and other signal readers. To use, select the property on the left to graph. You can allow the system to dynamically set its maximum and minimum values or you can select specific values. By default, you will see the values at the sample rate of the property.

When already graphing, re-open the graph dialog by pressing the gear-box icon. By default, graphing will

Using Widgets

Widgets are screen elements that allow you to setup a user interface to control your PHiL board. Buttons, status icons, lists, gauges and more can be placed on the screen and setup to monitor and control your input and output.

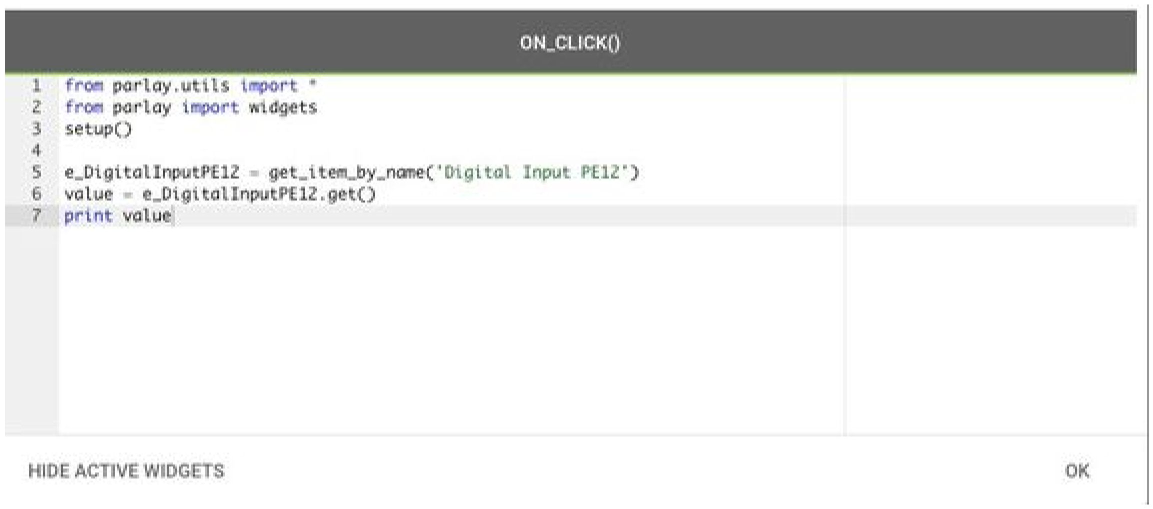
Widget Basics

Widgets are customizable controls for your dashboard that can be used to communicate with your items either independently or coordinated. Widgets are setup to either monitor specific item properties (such as to graph) or run a Python script when their widget specific event (ex: button push) occurs.

Creating a Widget from a command

The Python command code can copy and paste from the Item Card when the appropriate command is setup. You can also directly create a widget from the Item command by pressing the "+" button in the scripting area and selecting the button widget.

Commands that return data will be handled synchronously when using the Item script support. Below is an example of the Python to get a GPIO input.



```
from parlay.utils import *
from parlay import widgets
setup()

e_DigitalInputPE12 = get_item_by_name('Digital Input PE12')
value = e_DigitalInputPE12.get()
print value
```

Note that in the above example when we use "print" it will be displayed on the screen when executed.

If the command returns an array that becomes a list in python. Likewise if it returns a few different parameters they will be returned as a Python dictionary.

Sleep

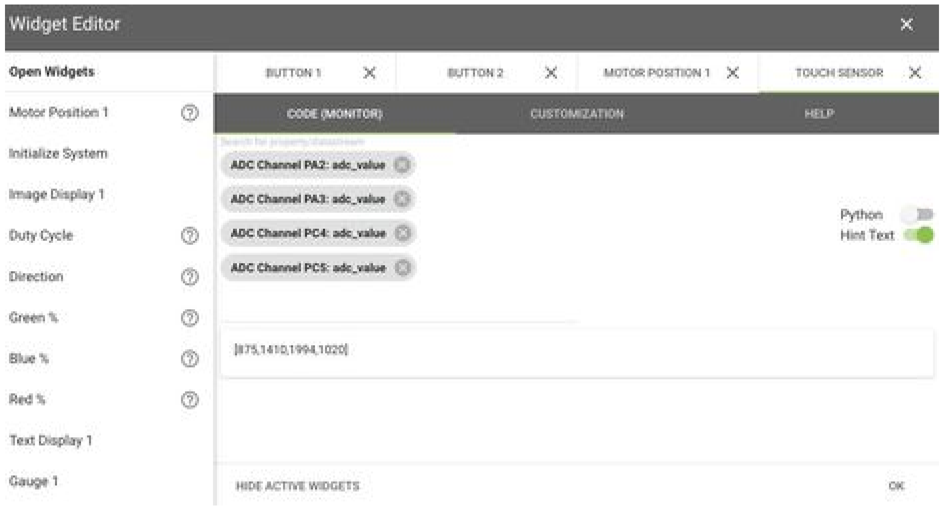
To put delays in the code the sleep() function is used. The units are in seconds.

Print

The print() function will cause the values printed to show up on the bottom right of the screen.

Monitoring Widgets

Widgets that monitor input can either directly monitor the property of the item or can use Python code. The example below is of the selection of multiple properties being setup to be monitored in the graph widget:



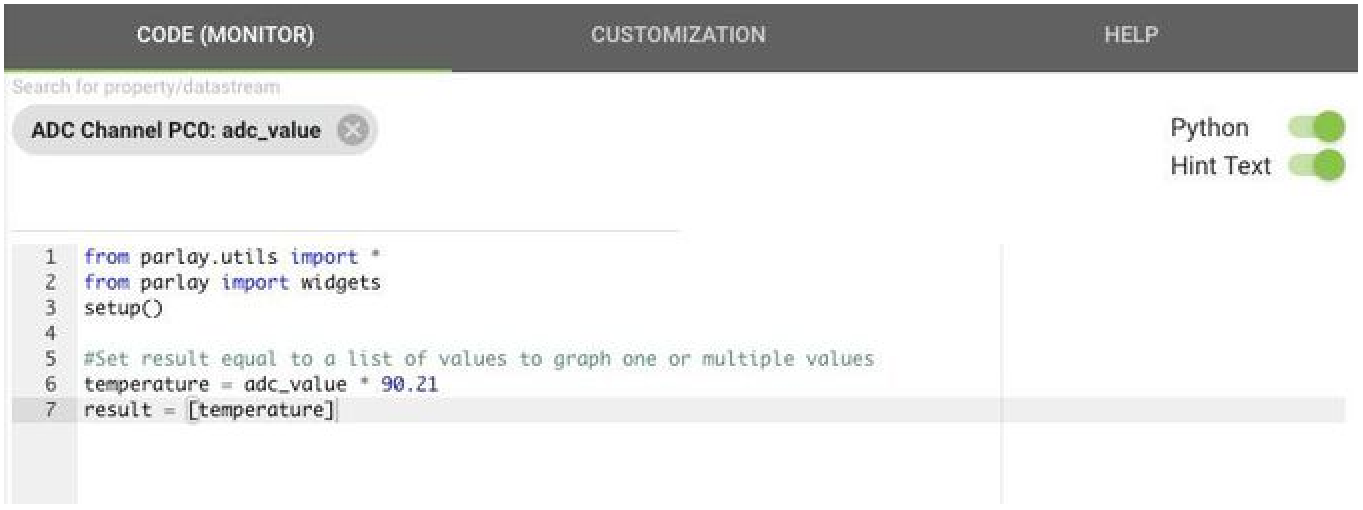
ADC Channel PA2: adc_value
ADC Channel PA3: adc_value
ADC Channel PC4: adc_value
ADC Channel PC5: adc_value

Python
Hint Text

```
[875,1410,1994,1020]
```

Note that in the above screen-shot, Python is disabled. If the data should be manipulated before being displayed, python can be enabled.

For example, if some math on the raw data needs to be performed it can be done in Python for the display.



```
from parlay.utils import *
from parlay import widgets
setup()

#Set result equal to a list of values to graph one or multiple values
temperature = adc_value * 90.21
result = [temperature]
```

Python Scripting

Parlay can be run from the UI or using Python 2.7 scripts. Via Python it is possible to create Items for peripherals not on PHiL. This could be useful for using and coordinating external equipment or using an API to another device.

Python scripting in Parlay is defined in the following link:

http://parlay.readthedocs.io/en/latest/index.html

Pin Control Specifications

This section describes I/O available and their features. For more details, refer to the STMicro STM32F407 datasheet: http://www.st.com/resource/en/datasheet/DM00037051.pdf

General Purpose IO (GPIO)

Inputs

The GPIO inputs are expecting to input 0.0v/3.3v for 0/1. The value is read-able from a command or from the *value* property. The *value* can be graphed, or monitored in any monitoring widget (gauge, charge...).

By default GPIO inputs have pull-up or pull-down configured. Weak pull-ups and pull-downs are available from the microprocessor and can be set via item command "set_pull".

GPIO Inputs have a sample rate property, which is the rate the pin is internally sampled. When graphing, or using any monitor type of widget (ex: gauge) tied to the "value" property, this is also the rate that the value is sent to Parlay, unless a rate is provided (as in the item graph setup).

Outputs

GPIO Outputs will drive 0v to 3.3v up to 8mA. The "Set" command will set to 3.3v and the "Reset" command will set it to 0v. The "value" property can be set from the properties tab.

ADCs

The 12 bit ADCs will read 0-3.3v as 0-4095 values.

ADC's have a sample rate property, which is the rate the pin is internally sampled. When graphing, or using any monitor type of widget (ex: gauge) tied to the "value" property, this is also the rate that the value is sent to Parlay, unless a rate is provided (as in the item graph setup).

DAC

The 12 bit DAC will output 0-3.3v when set 0-4095 as output.

PWMs

PHiL lets you configure a PWM frequency (hz) and duty cycle (%). The hardware allows multiple channels per timer and the frequency is determined at the timer level. Each channel on a timer can have a unique duty cycle set. This relationship is represented in the Item drop down menu where the timer is the parent of the PWM item.

For example:

PWM PE5 and PWM PE6 has the parent timer of Timer A. To set the frequency, open Timer A Item and set the frequency either via command or property. To set the duty cycle, open the individual PWM pins (ex: PWM PE5).

PWM Readers

There are 2 inputs to read the frequency and duty cycle of a PWM signal. The two readers are selected to serve potentially different needs. PWM Input PA1 uses a slower timer (84Mhz) and should not be used for frequencies greater than 100kh for accurate duty cycle readings. PE9 uses a faster timer (168Mhz) but because it uses a 16 bit timer, it should not be used for frequencies less than 5Khz without increasing the pre-scaler.

Readers have a sample rate property, which is the rate the reading is sampled from the timer and calculated. When graphing, or using any monitor type of widget (ex: gauge) tied to the "value" property, this is also the rate that the value is sent to Parlay, unless a rate is provided (as in the item graph setup).

Accumulators

The accumulators count the number of pin rises. PA0 is on a 32 bit timer. PF7 is on a 16 bit timer, and care must be taken to make sure that the sample rate is fast enough to not overrun the timer.

Encoder

The encoder reader can read the 2 pins of a quadrature encoder and provide relative distance information, automatically accommodating direction changes. For precise readings raise the sample rate up to 100Hz.

Encoders have a sample rate property, which is the rate the reading is calculated. When graphing, or using any monitor type of widget (ex: gauge) tied to the "value" property, this is also the rate that the value is sent to Parlay, unless a rate is provided (as in the item graph setup).

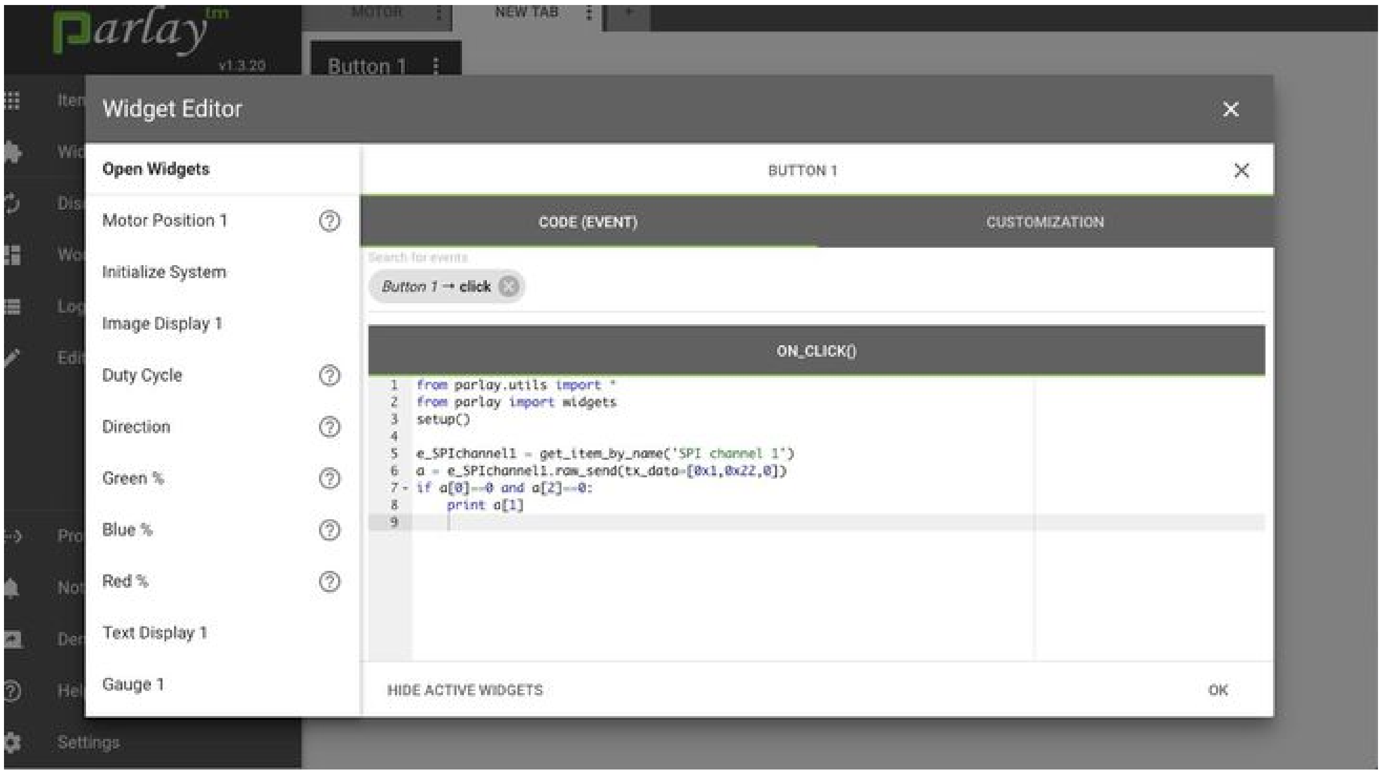
UART

A simple RX/TX Uart raw byte or character UART protocol is available. For data, type comma delimited data to send. Received data will be in a 20 byte cyclic data buffer. The rx_index points to the index of the next data to be received. The special 4 pin connector near the USB provides TX, RX for the UART.

SPI

Send and receive of raw SPI data is available. Connect the Chip Select pins for the given channel to the SPI device CS. You do not need to control the chip select - it will be driven low during the SPI transfer. Open the channel item card and type the comma delimited byte data to send.

Because you are sending raw data, a slightly easier way to handle SPI is with Python used within a button widget. Using Python can represent the data as Hex and the result is received an array that can be evaluated.



```
from parlay.utils import *
from parlay import widgets
setup()

e_SPIchannel1 = get_item_by_name('SPI channel 1')
a = e_SPIchannel1.raw_send(tx_data=[0x1,0x22,0])
if a[0]==0 and a[2]==0:
    print a[1]
```

Note that data is received as a result of the raw_send command, and the data is a list. Process the received data per your needs.

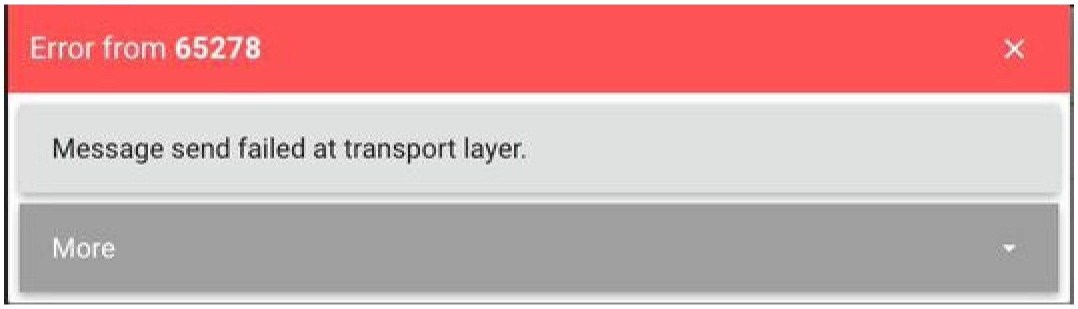
I2C

Like the SPI, raw data is easier to manage with the Python script than the item card.

Troubleshooting

Connection problems

The following error indicates that Parlay is unable to communicate with the PHiL board. This could be because the board is not connected at the time of start.


Error from **65278**

Message send failed at transport layer.

This is generally resolved with the following steps if you were previously connected:

1. Re-connect USB to your PC and board
2. Select "Protocols" from the left panel
3. "Disconnect" the PCOMSerial Protocol
4. "Reconnect" the protocol
5. Run Discovery from left panel.

To connect a board after Parlay has been started:

1. Connect USB to PHil to your PC.
2. Select Protocols from the left panel
3. Select PCOMSerial Protocol
4. For the port, select the one detected.
5. Run Discovery.

Display problems

PHiL is tested in Chrome and Chromium (from Windows installer). If you have installed Parlay by means of pip, you should run it in Chrome for best results

Sluggish Display

If you increase sample rate of Items and/or have a lot of data monitors you may experience delays in your browser application as it tries to process all the received data.

PHiL Pins

Left Pin Header

Left Pin	Left Pin Assignment	Right Pin Assignment	Right Pin
GND			5V
GND			3V3
PB12	GP Output	GP Output	PB13
PB14	rsvd	rsvd	PB15
PD8	GP Output	GP Output	PD9
PD10	GP Output	GP Output	PD11
PD12	GP Output	GP Output	PD13
PD14	GP Output	GP Output	PD15
PG2	GP Input	GP Input	PG3
PG4	GP Input	GP Input	PG5
PG6	GP Input	GP Input	PG7
PG8	GP Input	PWM	PC6

Right Pin Header

Left Pin			Right Pin
3v3			3V3
PB2			BT0
VREF			GND
PB10	GP Output	GP Output	PB11
PE14	N/A	N/A	PE15
PE12	GP Output	GP Output	PE13
PE10	GP Output	GP Output	PE11
PE8	N/A	PWM Reader	PE9
PG1	GP Input	N/A	PE7
PF15	N/A	GP Input	PG0
PF13	N/A	N/A	PF14
PF11	N/A	N/A	PF12

PC7	PWM	rsvd	PC8
PC9	rsvd	N/A	PA8
PA9	USART	UART	PA10
PA11	rsvd	rsvd	PA12
PA13	rsvd	rsvd	PA14
PA15	rsvd	rsvd	PC10
PC11	rsvd	rsvd	PC12
PD0	SPI CS	SPI CS	PD1
PD2	SPI CS	SPI CS	PD3
PD4	GP Output	GP Output	PD5
PD6	GP Output	GP Output	PD7
PG9	GP Input	GP Input	PG10
PG11	GP Input	GP Input	PG12
PG13	GP Input	GP Input	PG14
PG15	GP Input	SPI sclk	PB3
PB4	SPI MISO	SPI MOSI	PB5
PB6	Encoder	Encoder	PB7
PB8	N/A	N/A	PB9

PB1	rsvd	rsvd	PB2
PC5	ADC	N/A	PB0
PA7	Encoder	ADC	PC4
PA5	DAC	Encoder	PA6
PA3	ADC		PA4
PA1	PWM Reader	ADC	PA2
PC3	ADC	Accumulator	PA0
PC1	ADC	ADC	PC2
PF10	rsvd LED	ADC	PC0
PF8	N/A	rsvd LED	PF9
PF6	PWM	Accumulator	PF7
PF4	ADC	ADC	PF5
PF2	N/A	rsvd	PF3
PF0	I2C SDA	I2C SCL	PF1
PE6	PWM	rsvd	PC13
PE4	rsvd	PWM	PE5
PE2	N/A	rsvd	PE3
PE0	N/A	N/A	PE1