qcl

April 4, 2018

In [ ]:
```
import pyb


        class PulseGenerator:
            def __init__(self,
                         channel=1,
                         pin_name='A0',
                         duty_cycle=0.5,
                         timer=None,
                         freq=500,
                         tim=2):
                """ Generate a pulse width modulated signal on given pin

                         Keyword Arguments:
                             channel {int} -- PWM channel to use (default: {1})
                             pin_name {str} -- pin name to use (default: {'A0'})
                             duty_cycle {float} -- fraction of time the signal is on (default: {
                             timer {[pyb.Timer]} -- a timer object (default: {None})
                             freq {int} -- frequency for PWM (default: {500})
                             tim {int} -- timer to use (default: {2})
                """

                self.channel = channel
                self.pin = pyb.Pin(pin_name)
                self.duty_cycle = duty_cycle

                if timer is None:
                    self.timer = pyb.Timer(tim, freq=freq)

                elif type(timer) == pyb.Timer:
                    self.timer = timer

            def set(self):
                """Set the pulse width for the PWM signal
                """

                self.pulse_width = int(self.timer.period() * self.duty_cycle)
```

```
        self.ch = self.timer.channel(self.channel, pyb.Timer.PWM, pin=self.pin)
        self.ch.pulse_width(self.pulse_width)


class Controller:
    def __init__(self,
                 pin='A4',
                 currentmax=3.0,
                 vmax=1.5,
                 vref=3.0,
                 amp_gain=10,
                 shunt_resistor=0.05):
        """Class for different settings of QCL driver.

        Keyword Arguments:
            pin {str} -- pin name (default: {'A4'})
            currentmax {float} -- maximum current for a given QCL (default: {3.0})
            vmax {float} -- maximum voltage for a given QCL (default: {1.5})
            vref {float} -- reference voltage of the board (default: {3.0})
            amp_gain {int} -- amplifier gain (default: {10})
            shunt_resistor {float} -- low resistance precision resistor
                                      used to measure current (default: {0.05})
        """

        self.currentmax = currentmax
        self.vmax = vmax
        self.vref = vref
        self.pin = pin
        self.amp_gain = amp_gain
        self.shunt_resistor = shunt_resistor

    def current_setting(self, current):
        """Set the QCL current

        Arguments:
            current {[float]} -- QCL current value
        """

        self.current = current
        volt = self.current / self.currentmax * self.vmax
        dac_value = volt / self.vref * 4096
        dac = pyb.DAC(pin=self.pin, bits=12)
        dac.write(dac_value)

    def shutdown(self):
        """Safety function to shutdown the QCL if the settings are out of bounds
        """
```

```
            pass

        def feedback_loop(self):
            """Implements feedback-loop safety procedure
            """
            pass

        def is_safe(self):
            """Checks if the settings are in range. If out of bounds, calls shutdown()
            """
            pass
```

In [ ]:
```
# Create a pulse-width modulated signal on pin 'A0' with default settings
pwm = PulseGenerator()
```

In [ ]:
```
controller = Controller()
controller.current_setting(current=2.0) # Set a current of 2A
```