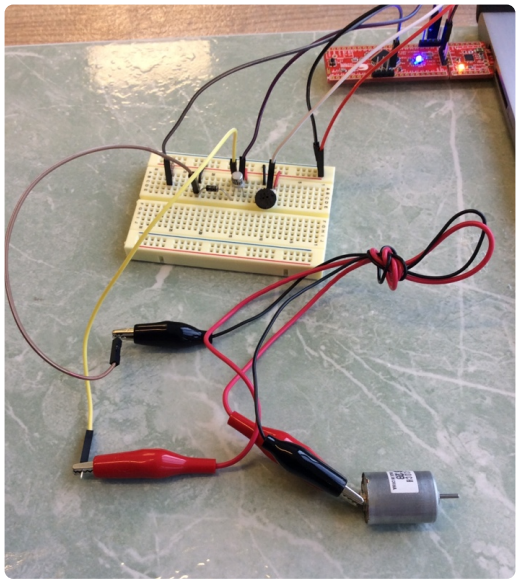
PSoC and PWM

Craig Cheney

V0.1
8/12/14

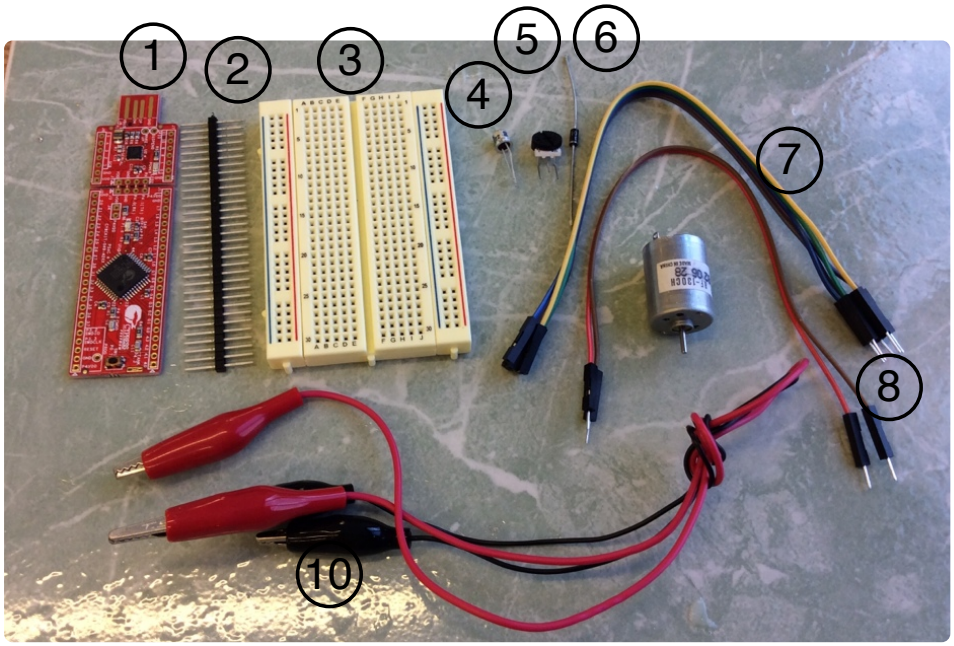
Objective: Use the PSoC 4 Prototyping Kit to create a low-cost, switching, motor controller.

[Final Picture - of working demo]

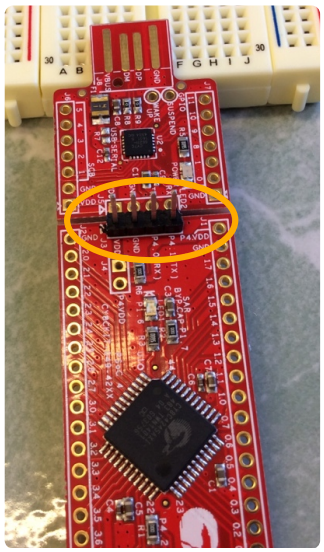


Materials list:

1. (x1) **PSoC 4** Prototyping Kit
2. (x14) **header** pins
3. (x1) solderless **breadboard**
4. (x1)1kΩ **potentiometer**
5. (x1) 2n2222 **transistor**
6. (x1) 1N4001 **diode**
7. (x4) Female - Male **jumper wires**
8. (x2) Male - Male **jumper wires**
9. (x1) **DC motor**
10. (x2) **Alligator Clips**
11. (x1) **Soldering iron**
12. (x1) 1kΩ **Resistor**

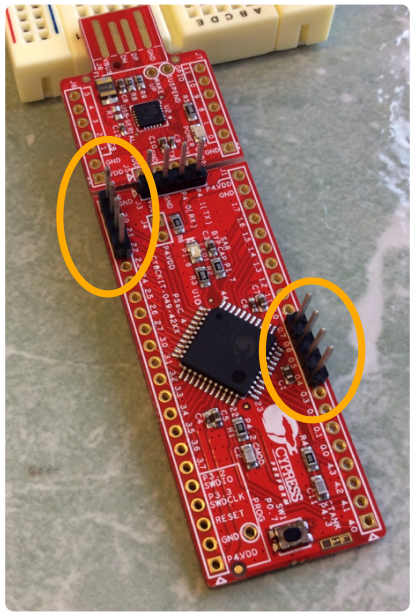


1. Solder the **header pins** into the **PSoC**
   a. Use a row of **4 headers** into VDD , GND, P4.0, and P4.1 at the top of the **PSoC** board.



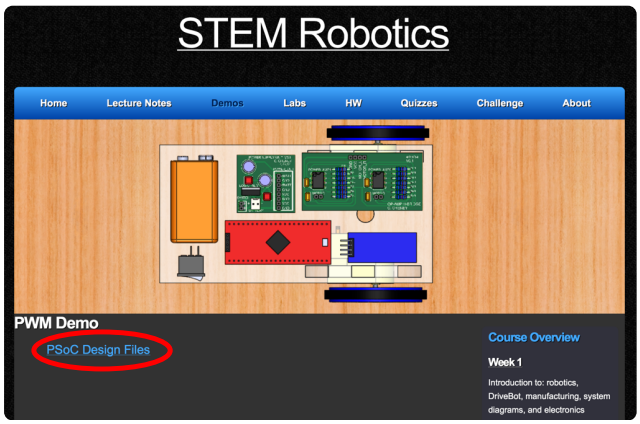
   b. use a row of **4 header pins** to solder into GND, 2.0, 2.1, and 2.2 on the top left side of the **PSoC**.

   c. Use a row 4 header pins to solder into 0.7, 0.6, 0.5 and 0.4 on the middle right side of the **PSoC**

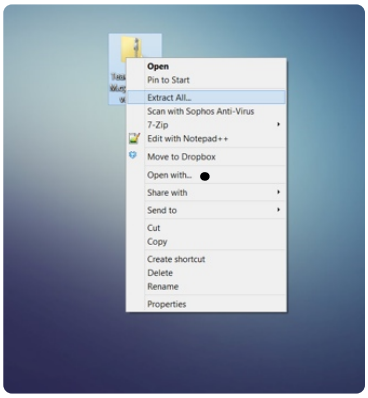


2. Download **PSoC Creator** if it is not already installed on your computer from http://www.cypress.com/psoccreator/
(Note: you must be on a Windows based computer)

3. Download the **Project** from:
http://stem.ccheney.net/demos/Teachers_PWM.cywrk.Archive01.zip
Make sure to save the folder to your **desktop**.

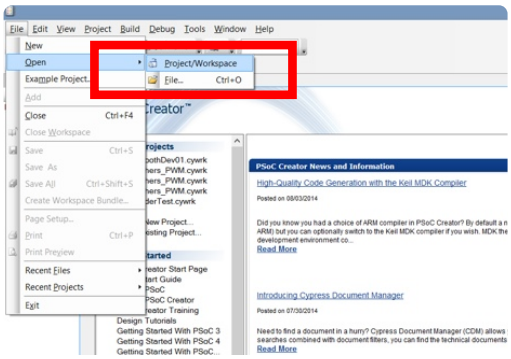


b. Unzip the folder by right-clicking and selecting **Extract All...**

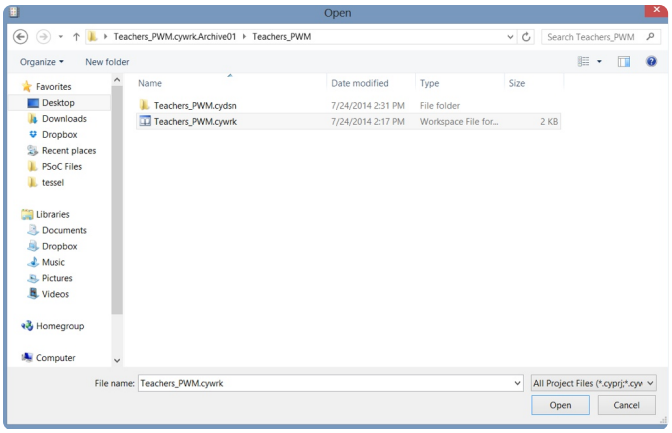


4. a. Open **Psoc Creator**

   b. In the home page of **Creator** navigate to File > Open > Project/ Workspace

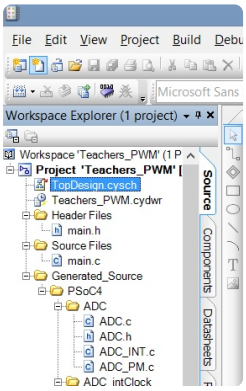


   c. **Navigate** to Desktop/Teachers_PWM.cywrk.Archive01/ Teachers_PWM/Teachers_PWM.cywrk

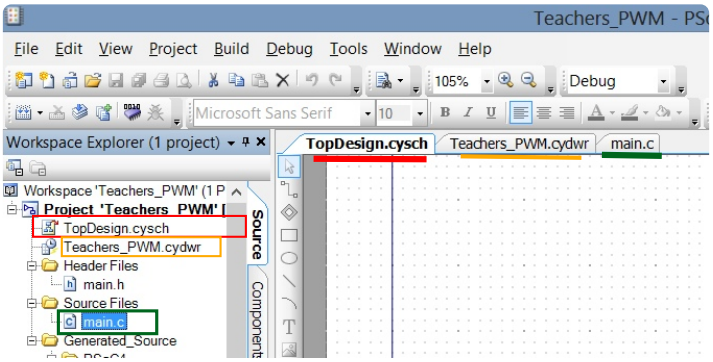


   d. hit **Open**

5. a. On the left hand side, **double click** TopDesign.cysch

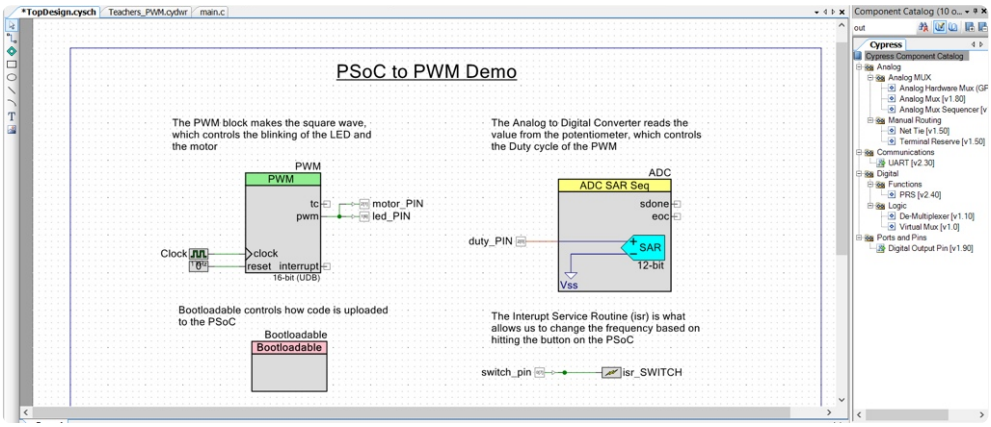


b. **Double click** on two more files, Teachers_PWM.cydwr, and main.c. You should now see three tabs at the top of the workspace, corresponding to the files you just opened.

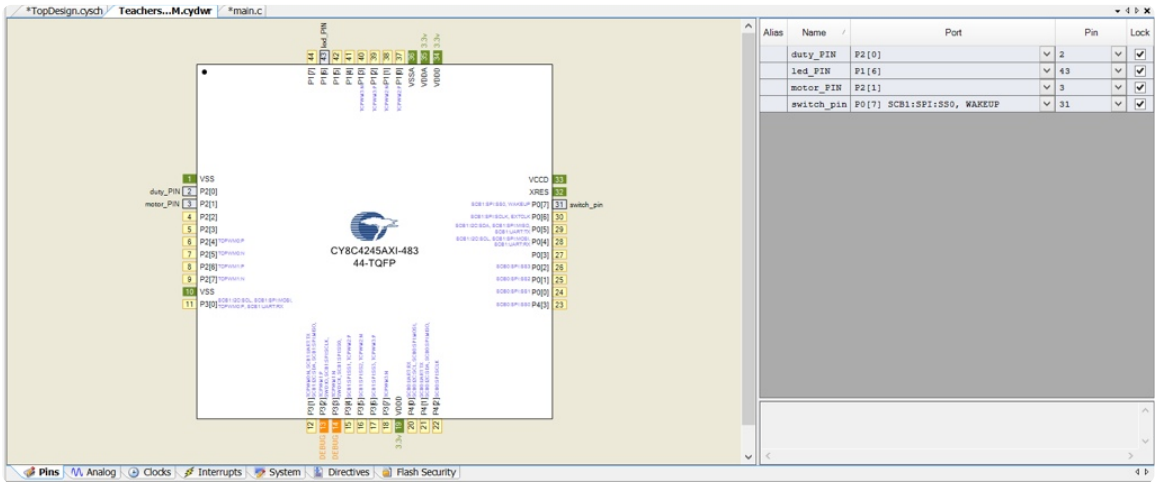


File Meanings

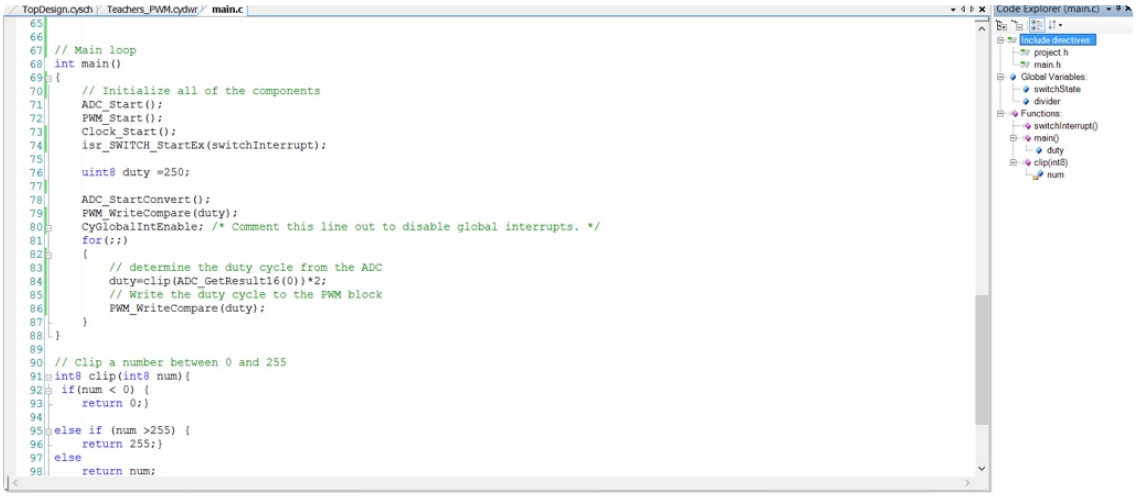
.cy**sch** - Cypress **Schematic** - This is a system diagram for your file. Components are dragged down from the component catalog on the right. These "virtual" components define most of the functionality of the file



.cy**dwr** - Cypress Design Wide Resources- This is the part of the file where the user connects the "virtual" components in the Schematic to the hardware. For example, in this tutorial we want to have the LED_pin turn on the led on the PSoC board, which happens to be at Port 1.6.

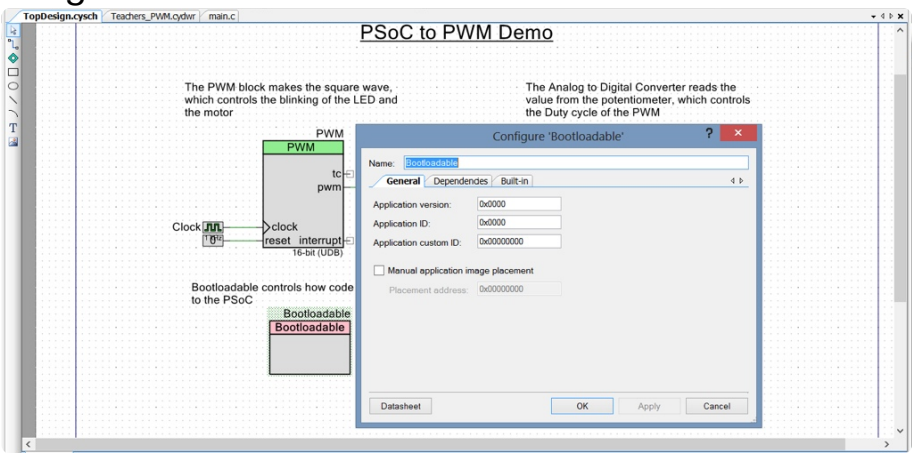


main.**c** - the C Language file - This is where the user written code is placed.

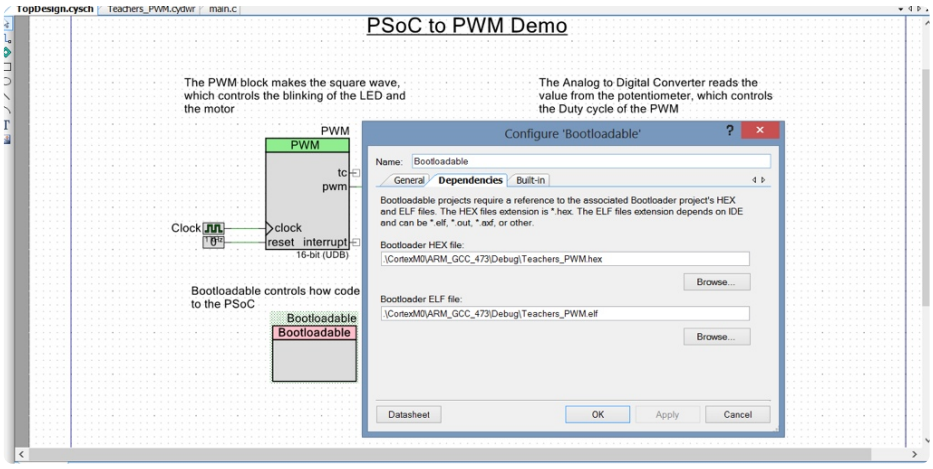


With the .cydwr, .cysch, and .c files, the user can completely control the functionality of the PSoC.

6. First, we need to Upload code to the **PSoC**.
    a. Goal: Configure the **Bootloadable** block so code can be uploaded
        1. Navigate to the **Schematic File**
        2. Double click the **Bootloadable** block to bring up the configurations menu

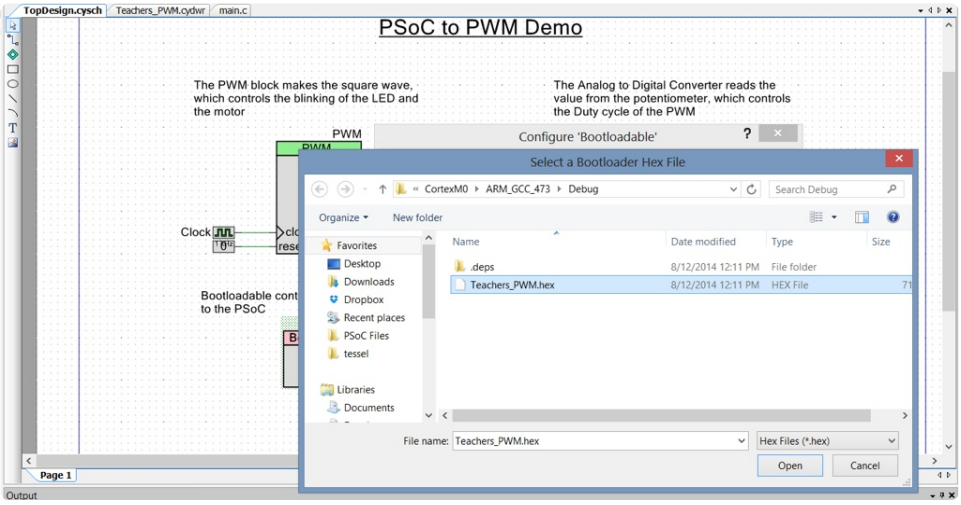


        3. Click **Dependencies**



        4. Under "Bootloadable HEX file:" click **Browse...**
        5. Navigate to: **\Desktop \Teachers_PWM.cywrk.Archive01\Teachers_PWM \Teachers_PWM.cydsn\CortexM0\ARM_GCC_473\Debug \Teacher_PWM.hex** *

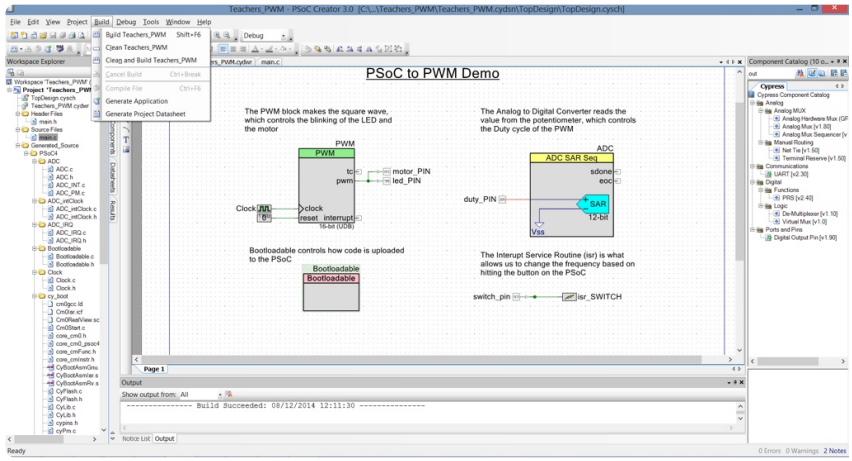
See note 1.



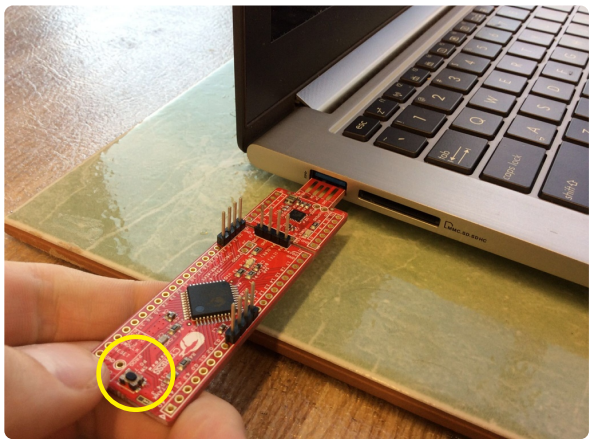
6. Click **Open**
7. Click **OK**

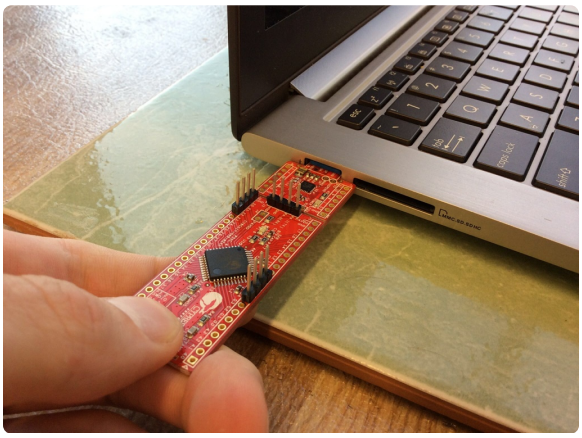
b.Goal: Build the project so it is ready to upload to the PSoC

1. Click **Build** > **Build Teachers_PWM**

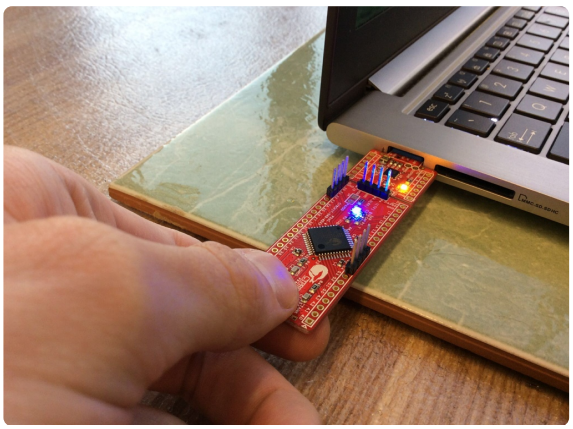


2. Insert the **PSoC** into the **USB port** while depressing the **button** at the opposite end.



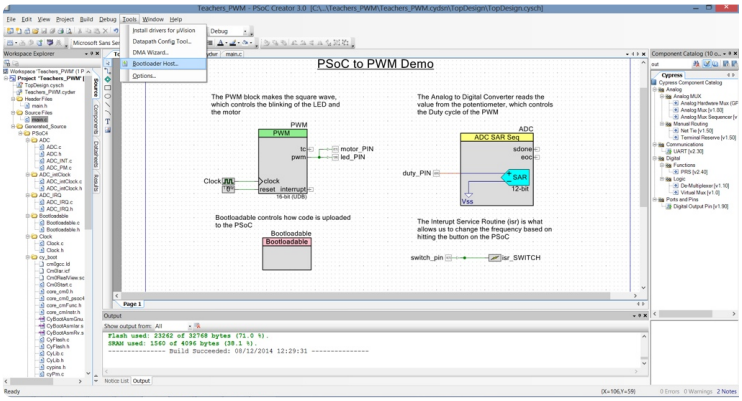


3. Holding the button signifies that the PSoC is in **programmable** mode. If done correctly you should see two lights, a *solid* **Orange Power LED** and a *blinking* **Blue Program LED**

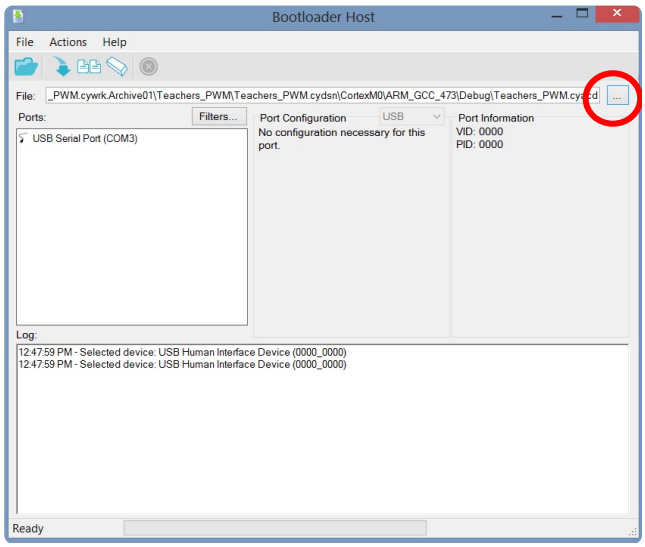


4. You can now let go of the **button** at the end of the PSoC

5. In Creator, navigate to **Tools** > **Bootloader Host..**



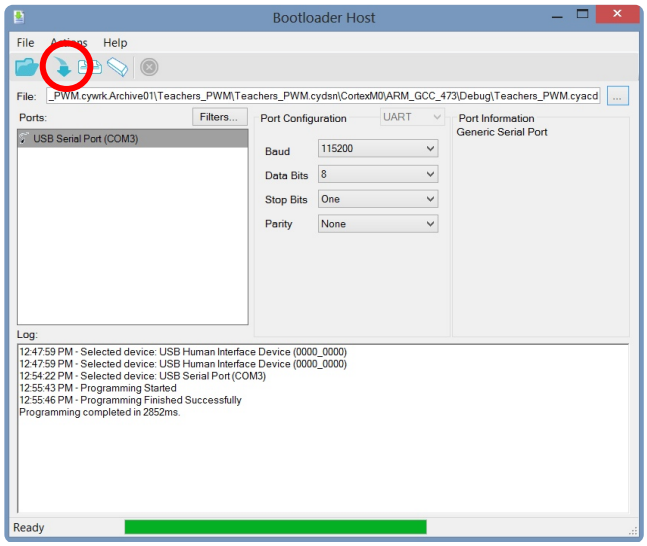
6. In the Bootloader Host, click **"..."**



7. Navigate to the bootloader file, located at: **\Desktop \Teachers_PWM.cywrk.Archive01\Teachers_PWM \Teachers_PWM.cydsn\CortexM0\ARM_GCC_473\Debug \Teacher_PWM.cyacd**

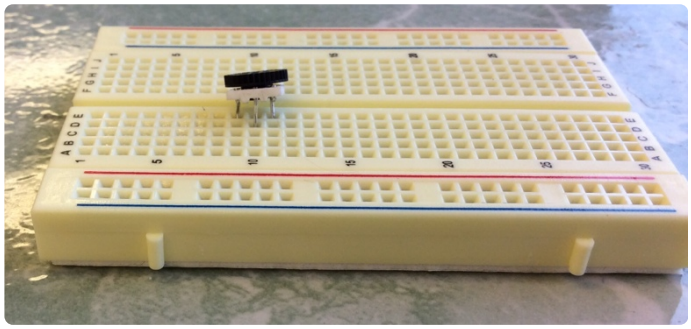
8. Click **Open**

9. Program the PSoC by clicking the down arrow, **Program**

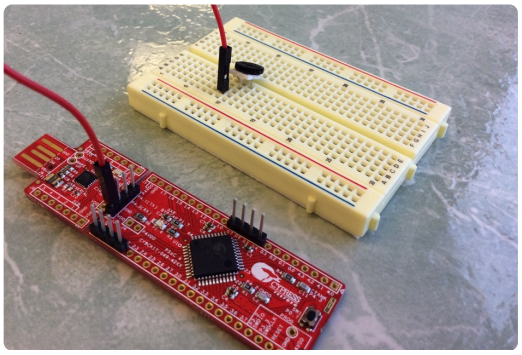


10. If the upload is successful, unplug the **PSoC**

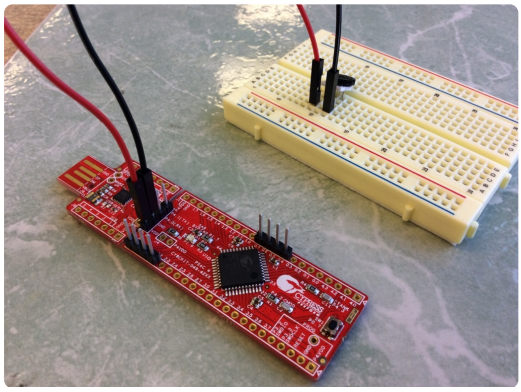
7. Now we're ready to connect the **circuit**
   a. Insert the **potentiometer** into the **breadboard**. *Note: the three legs of the potentiometer must be in 3 separate rows!*



   b. Using a **Male to Female Jumper wire** (preferably a red one), connect **VDD** on the PSoC to the **top leg** of the Potentiometer

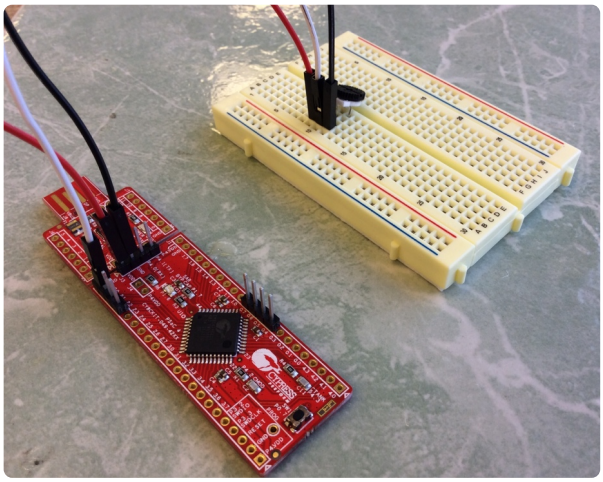


c. Using a M/F jumper wire (preferable a black one) connect **GND** to the **bottom leg**

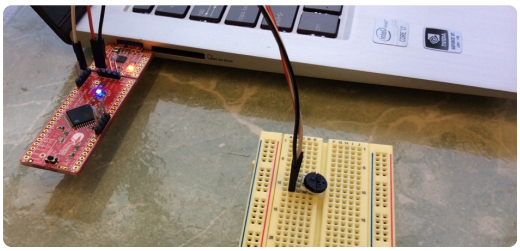


*Note: The convention with wire color is that **Red is power**, and **Black is ground**. Besides these two, the color of a wire makes no difference.*

d. Connect **P2.0** to the **middle leg** of the potentiometer



e. Plug the **PSoC** into the **USB port** *without* the button depressed (run mode)



8. The blue LED should now be blinking. Spin the potentiometer. What happens? This is Pulse Width Modulation! With the potentiometer knob all the way CounterClockwise (CCW) the Blue LED should be all the way on. With the knob all the way Clockwise (CW), the light should be all the way off. In between the two, it should be a varying mix of on and off.

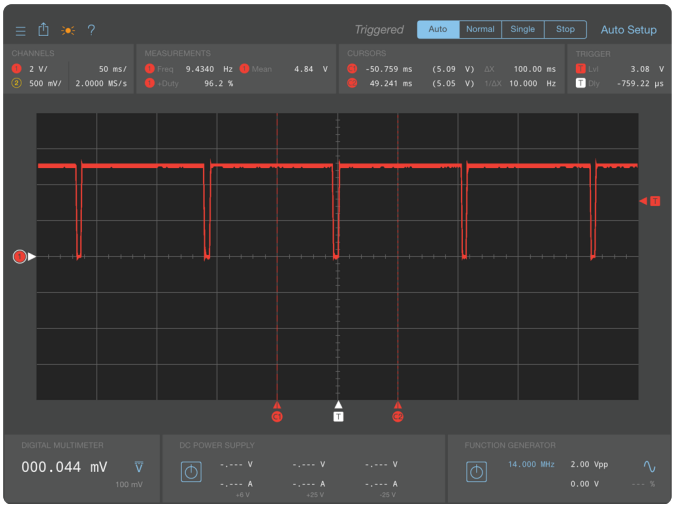
9. Let's that a look at the electrical signal being generated. With the know mostly CW (but now all the way) the light should blink on for a short period of time, but then be off for longer.

b. Notice the measurements in the photo above. The LED switches between 0 volts and 5 volts, 9.34 times per second, which is visible to the human eye. The number of time a PWM wave oscillates per second is called the **Switching Frequency**, and has units of Hertz (1/seconds, abbreviated Hz). F_switch = 9.34 Hz here.

This particular wave is high for 12.2% of the time, and low for (100%-12.2%) = 87.8% of the time. The percentage of the time spent high is called the **Duty Cycle**. D = .122 here.

The **mean value** can be calculated by multiplying the Duty Cycle by the "on" value. For us, V_on = 5 volts. D*V_on = Mean. Do these numbers work in our case? (Allowing for some amount of error).

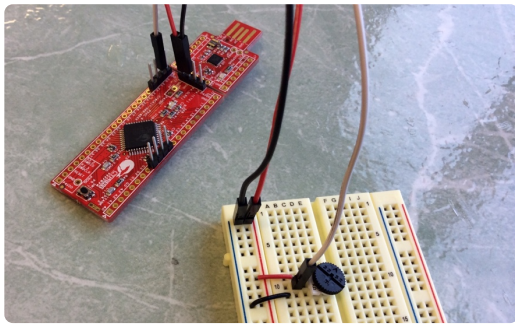
Look now at a signal that has a high duty cycle, so the knob is most of the way CCW. Here the light is on significantly longer than off, but it is still visibly blinking. Do the values for D, and the mean make sense? Did F_switch change?



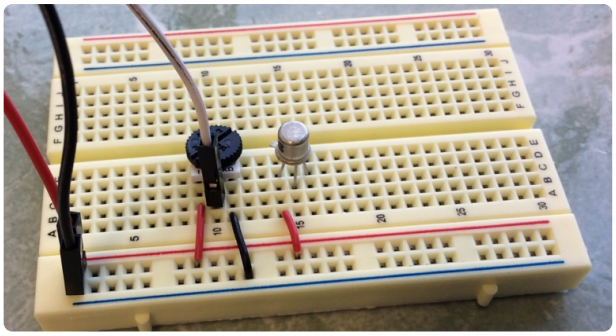
By changing the position of the potentiometer, we are changing,or modulating, the width of the high part, or the pulse. Hence, the name **Pulse Width Modulation**.

Let's hook up a Motor

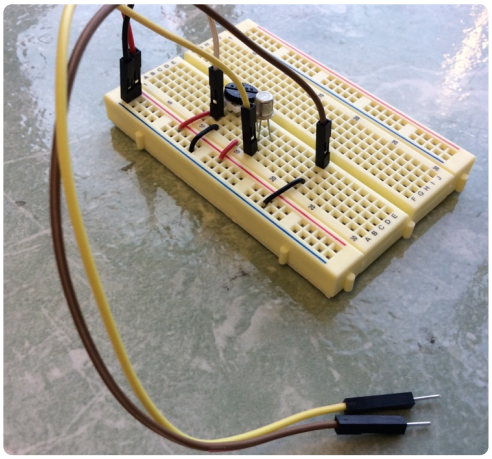
10. Goal - get our motor spinning, using PWM.
    a. Build a single transistor amplifier
        1. Modify the **existing circuit** to use the rails. You can use M/M jumpers, or bits of wires like shown. *Note: this circuit is functionally equivalent to the previous one.*



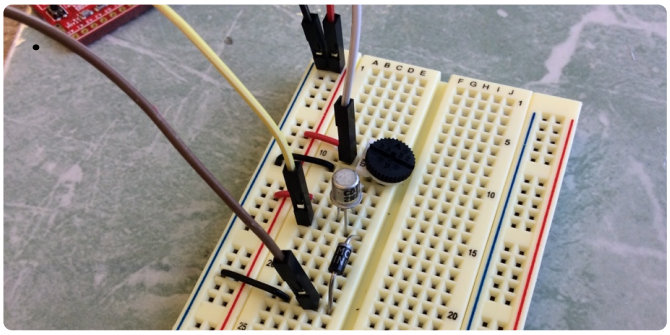
2. Insert the **2N2222 Transistor** into the breadboard, with the tab facing the upper right.
3. Connect a red wire from **power** to the **top leg** of the transistor



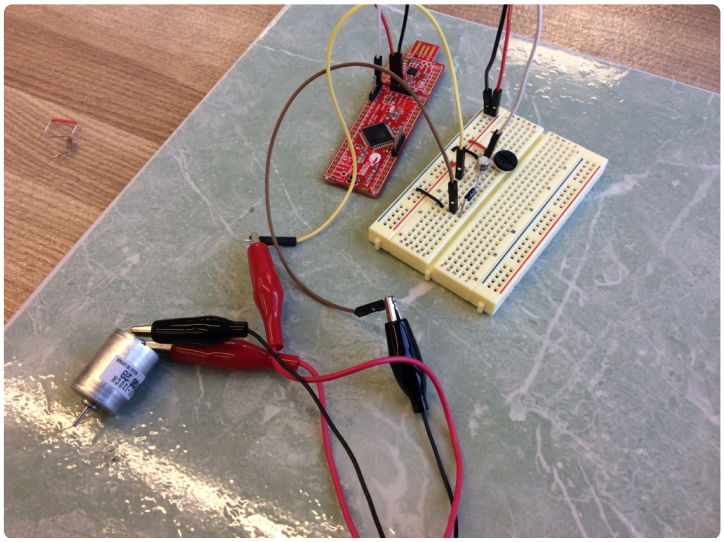
4. Connect a M/M jumper wire to the **bottom leg of the transistor.** Leave the **other end dangling** for now.
5. Connect a M/M jumper wire to **ground**, ~5 rows down the board. Leave the **other end dangling**



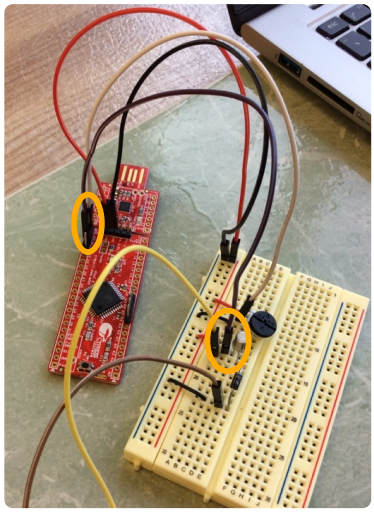
6. Insert the **1N4001 Diode**, connecting the two dangling wires, *with the white band facing the transistor.*



7. Using **two alligator clips**, connect the two **dangling jumper wires**, to the two **terminals of the motor**



b. Using a M/F jumper, connect the **middle leg** of the transistor to **P2.1** on the PSoC



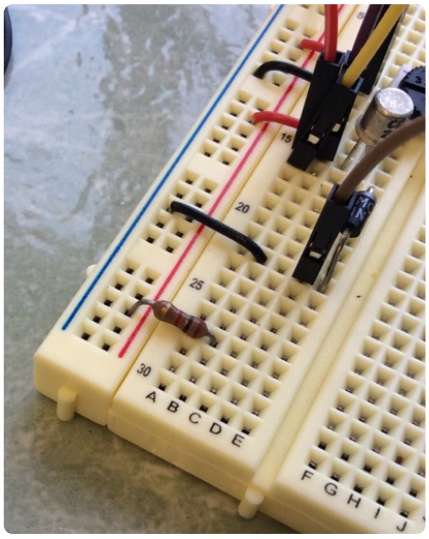
c. Plug the **PSoC** back into the **computer**. The motor should spin when the LED is on, and stop when the LED is off.

The duty cycle of the PWM, which corresponds to the position of the potentiometer, now controls the speed of the motor. However, if you hold onto the shaft, you can feel the motor **pulsing** on and off.

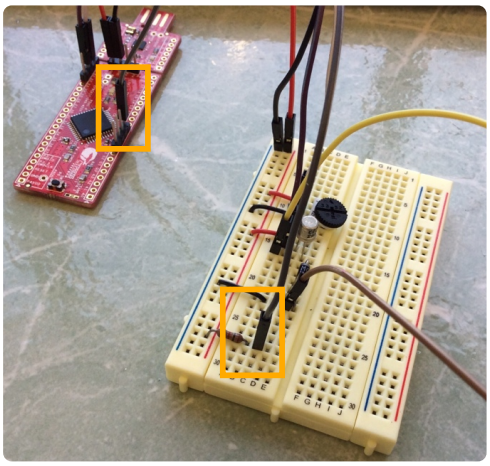
Pulsing in our motor is not a desirable effect, rather we want to see it **smoothly** changing the speed. The solution to this: **change F_switch** on the PSoC.

11. Goal : Have the PSoC change the switching frequency of the PWM by pressing the button on the PSoC.

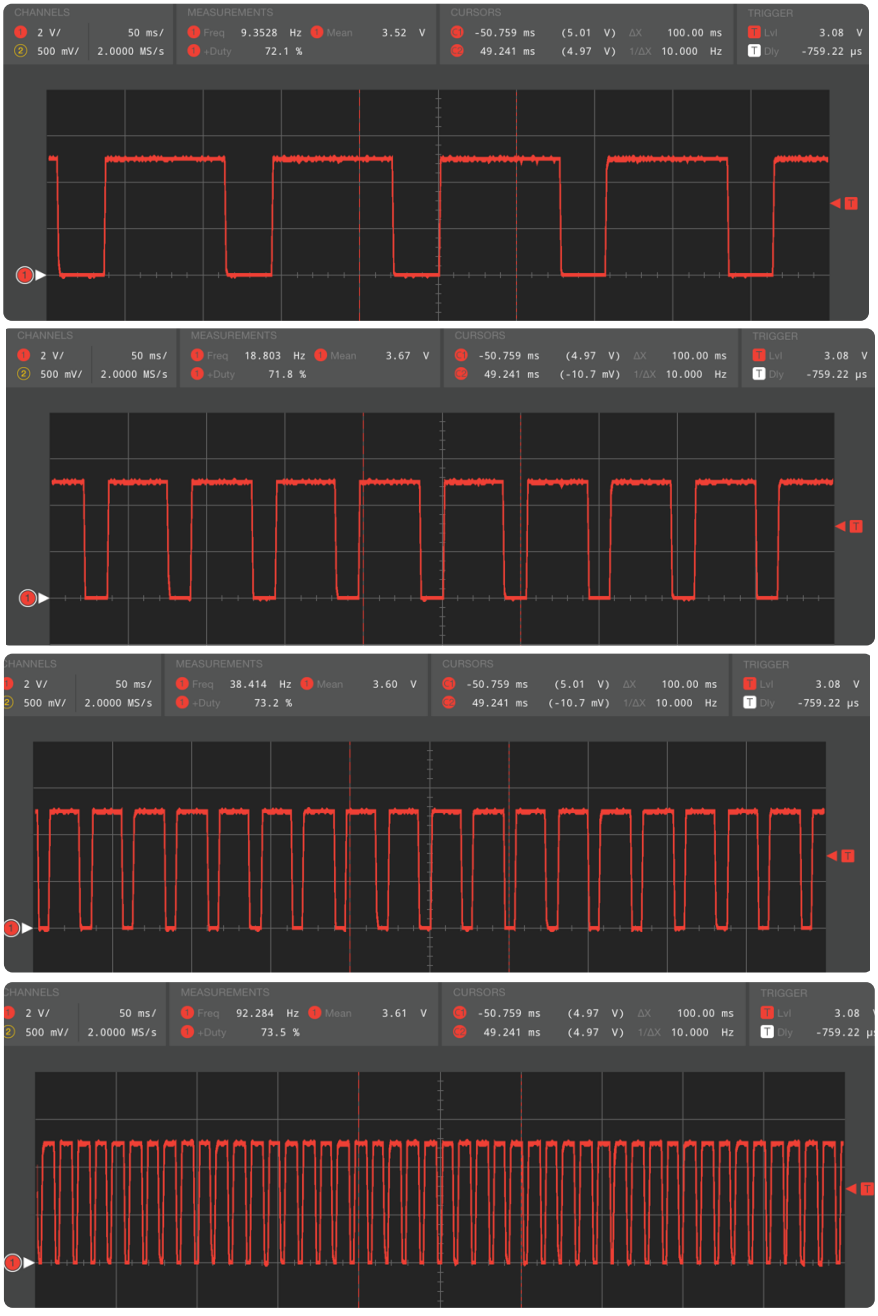
a. Wire up the button. Insert the **resistor** into the breadboard, with one leg in **power**, and the other connected to nothing.



b. using a M/F jumper wire connect the **open end** of the resistor to **P0.7** on the PSoC



c. Plug the PSoC back in. Press the **button** at the end of the PSoC. The Blue LED should blink quicker, while the duty cycle stays the same. Press the button again to increase the frequency. It will cycle through four frequencies, 9 Hz, 18 Hz, 38 Hz, and 92 Hz.



Notice how at the higher frequencies, your eye is unable to distinguish the pulsing blue light, it just appears as though the light is dimming as you decrease the duty cycle.

Conclusion: In this demo you learned

1. The basics of PSoC Creator
2. How to Upload code to the PSoC 4 Prototyping Kit
3. Pulse Width Modulation
    1. What the Duty Cycle is
    2. What the Switching Frequency is
    3. How to use a potentiometer to control the duty cycle
    4. How to use a button to control the Switching Frequency
4. How to build a single transistor amplifier
5. Basics of integrating a DC motor

For more information and tutorials please visit:
stem.ccheney.net