- Make an outline for what we should present at our last meeting - what have we worked on this semester, what worked/failed, gather some images and maybe make a quick slideshow

- The plan going forward is to use the new RTC module to make an alarm signal once per hour to turn on the power to the pico, so the code on the pico will just measure and log once then stop. But the alarm signal from the RTC doesn't last long enough, so we can use a 555 timer in monostable mode to make a longer pulse which will then go to the gate of a MOSFET that controls the power supply of the pico. Here's the steps towards that:

1. If you haven't a 555 timer, do some quick research to understand what it is. Wire up the 555 timer IC (datasheet) like in the Figure 3 below (but also include a pullup on the trigger pin and the Reset pin) for monostable operation:

8.4.1 Monostable Operation

In this mode of operation, the timer functions as a one-shot (Figure 3). The external capacitor is initially held discharged by internal circuitry. Upon application of a negative trigger pulse of less than 1/3 V_S to the Trigger terminal, the flip-flop is set which both releases the short circuit across the capacitor and drives the output high.

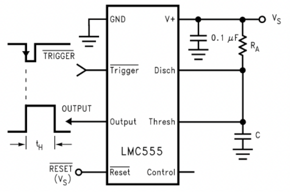

Figure 3. Monostable (One-Shot)

The voltage across the capacitor then increases exponentially for a period of $t_H = 1.1\ R_AC$, which is also the time that the output stays high, at the end of which time the voltage equals 2/3 V_S. The comparator then resets the flip-flop which in turn discharges the capacitor and drives the output to its low state. Figure 4 shows the waveforms generated in this mode of operation. Because the charge and the threshold level of the comparator are both directly proportional to supply voltage, the timing internal is independent of supply.

   Basically, when the trigger pin goes low, the output will go high for a period of time t=1.1*R*C (read the text below Fig. 3) and we want a time of roughly 1-10 seconds so I think R=100K and C=10uF would work. If you want to observe the output, put an LED with resistor on the output pin and change the state of the trigger pin to trigger it.

2. Use the RTC module as a trigger for the 555 timer: The DS3231 has two "alarms" you can set to go off at a specified time. When the alarm "goes off" it pulls the SQW pin low for less than a second. We want to use this to trigger the 555 timer.
   Install the DS3231 arduino library, read the documentation about the alarms so you know what it can do, connect the module to your Arduino's power/gnd/i2c pins(don't forget pullups for SDA and SCL), and upload the example sketch called "DS3231_set" (File->Examples->DS3231->D23231_set) and to get it to set the alarms (you have to send something in the serial console to set the RTC time I think, just read lines 27 and 100-109 in the code). Once the RTC is set with alarms, you can use the other example sketch "DS3231_test" to monitor the time and alarm status in the serial console. Or connect an led with resistor to the SQW pin. Notice that the output of the alarm is active low, just like the trigger pin on the 555 timer, so connect the alarm output to the trigger pin of the 555 timer.

3. Use a MOSFET to control the power of the pico: The pico can be powered without the usb connector by connecting 5V power to its VBUS pin (and GND to a GND pin). So check out the diagrams on this page, but just replace the light bulb with the pico and the Arduino with the output of the 555 timer. You may want to program the pico to have its onboard LED on so you know when it's powered.

4. With the RTC, 555 timer, and pico all wired up, the end result should be that when an alarm goes off, the pico is powered for ~5 seconds then turns off completely.

5. Edit the most recent pico code to only run once because we don't need it to sleep and loop now.