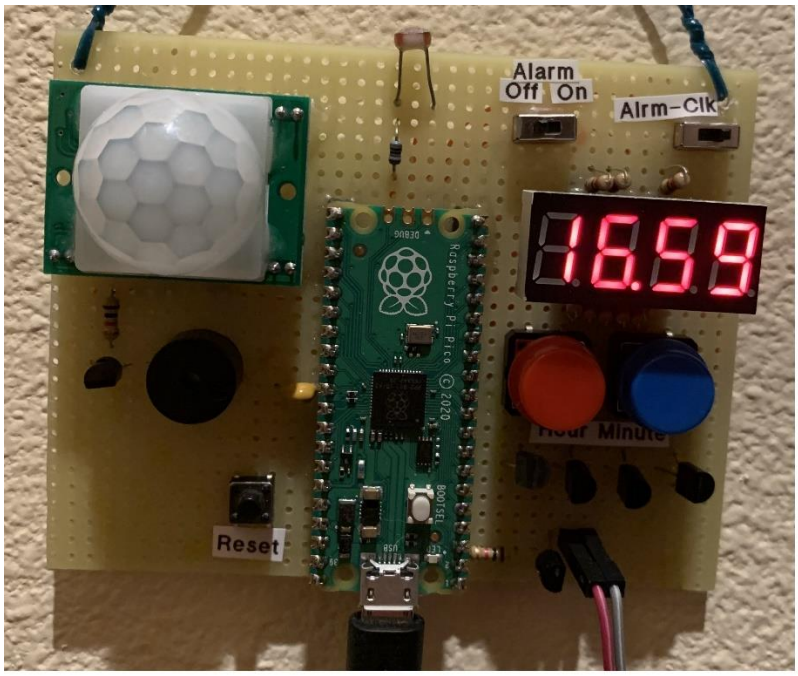
Raspberry Pi Pico LED Alarm Clock

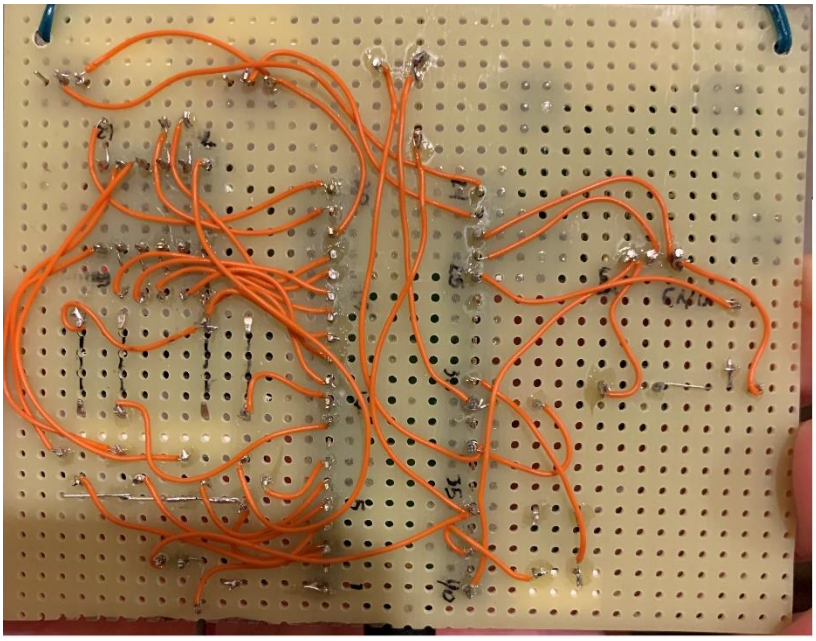


Raspberry Pi Pico with header pins
Female socket for Pico
4 Digit 7 Segment Common Cathode LED Display
PIR Sensor
Photo resistor
2 Slide Switches
2 Large push button switches
1 small push button switch
Passive Buzzer
Bright White LED
4 BS170 N Channel FETs
2 S8050 NPN Transistors
Resistors and Capacitors as described below

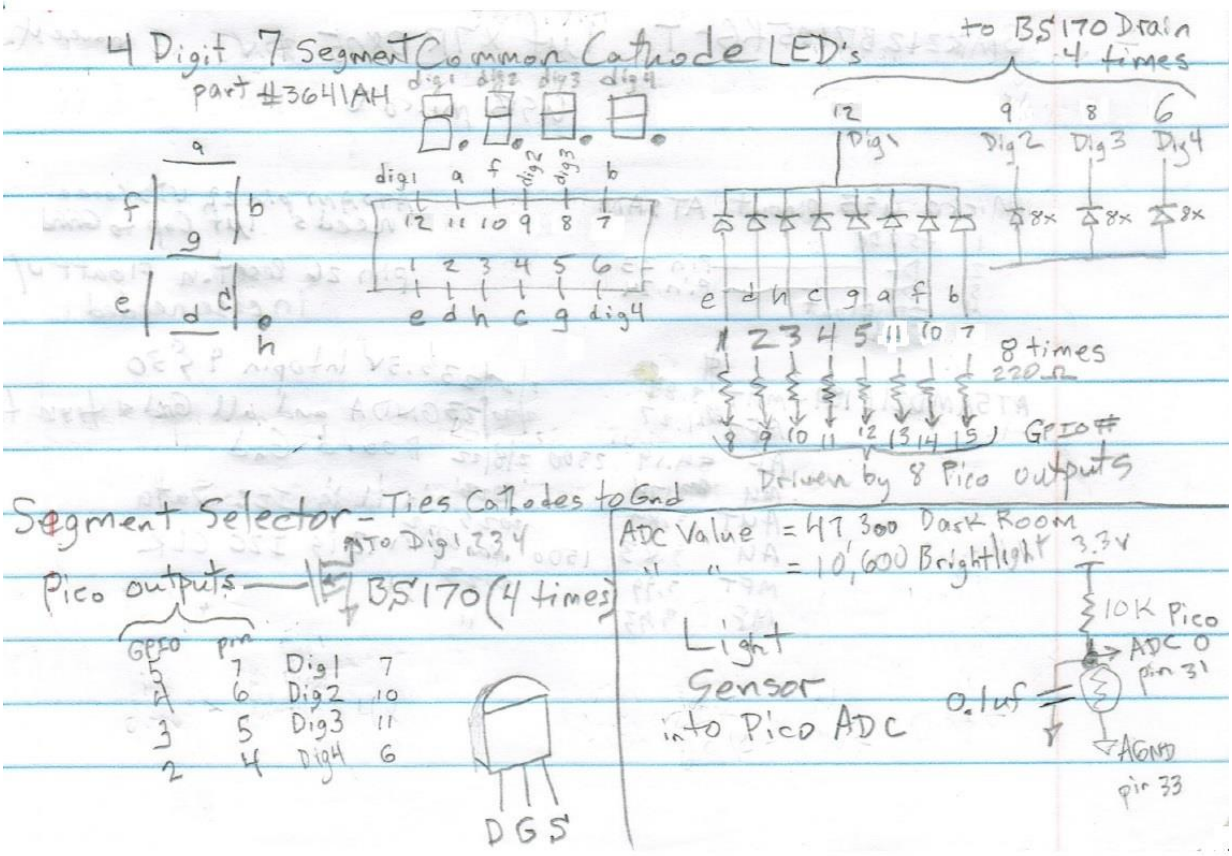
Introduction: I needed an LED alarm clock that only shows the time if there's motion, otherwise I want the room dark for sleeping. To solve this, I used a Raspberry Pi Pico, programmed with MicroPython to display the time on a 4 digit, 7 segment LED in 24 hour format. The display is active if a photo resistor detects the room lights are on or if a passive infrared sensor detects movement. A slide switch selects system time or alarm time and the hour and minute values can be incremented with two push button switches. A second slide switch turns the alarm on or off. A bright white LED is used as a night light and is turned on if the room is dark and movement is detected. When the system time matches the alarm time and the alarm slide switch is in the "On" position, a passive buzzer plays the intro to Beethoven's 5th symphony. A reset push button switch is connected to the Pico to gain control for re-programming. The code was renamed "main.py" and loaded into the Pico using the Thonny IDE so the Pico can run from a USB wall supply. The circuit board hangs low on the wall next to my bed.

Code: The MicroPython code for this project is called “alarm_clock_rev?.py” and can be found at my repo. This is my first Pico project and my first time programming in Python so expect ugly code that I’ll continue to revise as I gain experience. I found a Python example for controlling a 7 segment display that uses a 2 dimensional array but my limited experience needs to improve before using this method. Another area for improvement would be to use the Pico’s real time clock. Currently I’ve programmed an interrupt every second that increments the counters. It seems to keep good time but will probably need correction after a few months. The bright white LED nightlight is controlled using a PWM output from the Pico but it’s too intense so I’ve got the pulse width turned way down. The passive buzzer plays a tune when the alarm time equals the current time, then it stops. It’s loud enough to wake me so I don’t need it to repeat the tune or wait for me to push a button to stop the sound. The LED on the Pico shows if the Alarm switch is turned on.

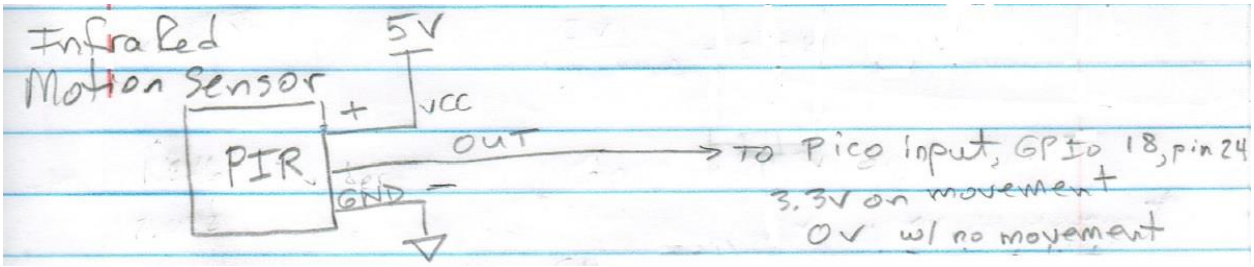
Build: Many of the parts in the list above are from an Adeept Ultimate Start Kit for a Raspberry Pi and can be purchased individually from Amazon and other sites. The build uses a prototyping vector board cut to size with sockets and parts held in place using J-B Weld epoxy. Wiring is soldered point to point using 30 gauge wire wrap wire as shown below.



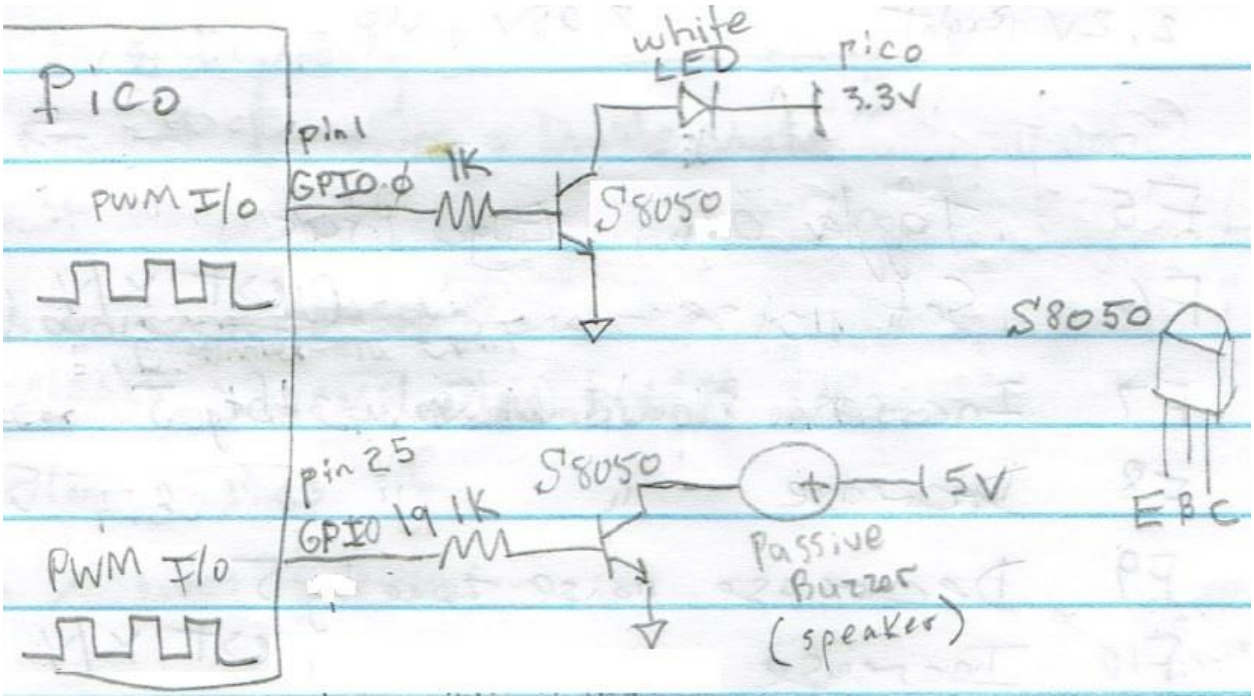
Schematics: The image below from my notebook shows the connections to the 4 digit 7 segment common cathode LED display, part number 3641AH from the Adeept kit and Amazon. The common cathodes for each segment are tied to ground using four N channel BS-170 FETs. The current for 7 segments and the decimal point combine and go thru the FET instead of using a bare Pico output that might be overloaded. The Pico drives the FET gate for each digit high, one at a time. The 8 anodes for the LEDs are driven high by the Pico thru 220Ω current limit resistors to light each segment. I could have used 4 resistors on the cathodes instead of 8 on the anodes but I wanted the segments to always have the same current. With my circuit, the brightness doesn't change when driving a "1" (2 segments) compared to an "8." (8 segments).



The schematic above shows the photo resistor from the Adeept kit and Amazon. It is tied to a 10KΩ resistor to make a voltage divider from the 3.3 volt supply. This voltage divider has a 0.1uf filter capacitor and is fed into the A to D converter of the Pico. I initially coded the Pico to print the ADC value on the Thonny console window to get the range. A bright room gave an ADC value of 10,600 and a dark room gave 47,300. I set the trip point in the code to 33,000 and it seems to work for my lighting. During the day or if I turn on the room light, the clock displays the time. If the room is dark and the passive infrared (PIR) sensor does not see any movement, the clock display is off. My PIR sensor is from the Adeept kit but many similar versions are sold. Some versions are powered with 3.3 volts but mine is powered with 5 volts (as shown below) and outputs a 3.3 volt signal when motion is detected. I've adjusted the two potentiometers on the PIR sensor to get the desired sensitivity and to hold the output signal active for several seconds so I have time to read the clock display if I hang my head over the side of the bed. The 5 volt power is from the VSYS pin on the Pico.

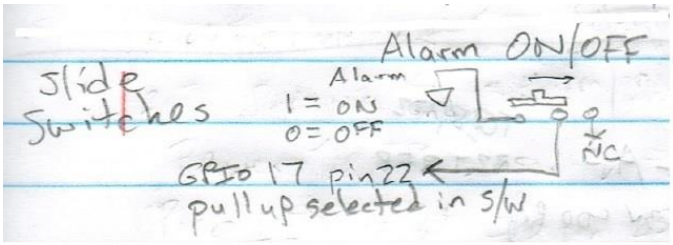


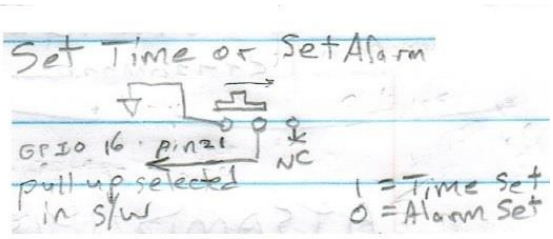
In addition to turning on the clock LEDs when the lights are low and movement is detected, a night light is also turned on. This bright white LED takes 30ma at 3.3 volts and is from an LED assortment kit by elegoo.com. The schematic below shows I used an S8050 NPN transistor (from the Adeept kit and Amazon) to switch the LED instead of driving it directly with the Pico output. The LED can handle 3.3 volts so no dropping resistor is needed. There is a 1KΩ resistor to limit the base current of the transistor.

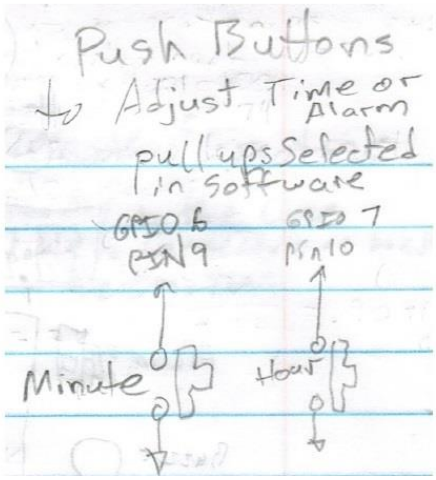


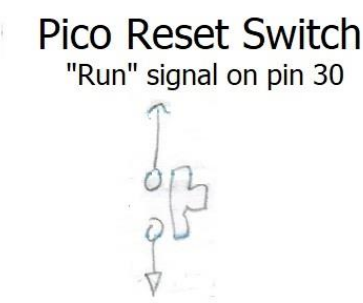
The passive buzzer from the Adeept kit (also found on Amazon) is basically a small speaker that measures about 15 ohms. To make sound, it must be driven with a PWM output from the Pico that is buffered with an S8050 NPN transistor. This transistor avoids overloading the Pico output with 333ma. The 5 volts is from the VSYS pin on the Pico.

The schematic below shows two slide switches are used to send ground or float an input signal into the Pico. The Pico software turns on the pull up for these GPIO inputs so the code can read a high or low. One switch selects whether the alarm is on or off and if on, the LED on the Pico is lit up. The other switch selects whether the display is showing the clock or the alarm time. Two push buttons under the hour and minute LEDs send high or low signals to the Pico so the software can increment the counters. These inputs are also programmed to have pull ups. I found it necessary to install a reset push button switch so I could gain control of the Pico for re-programming.

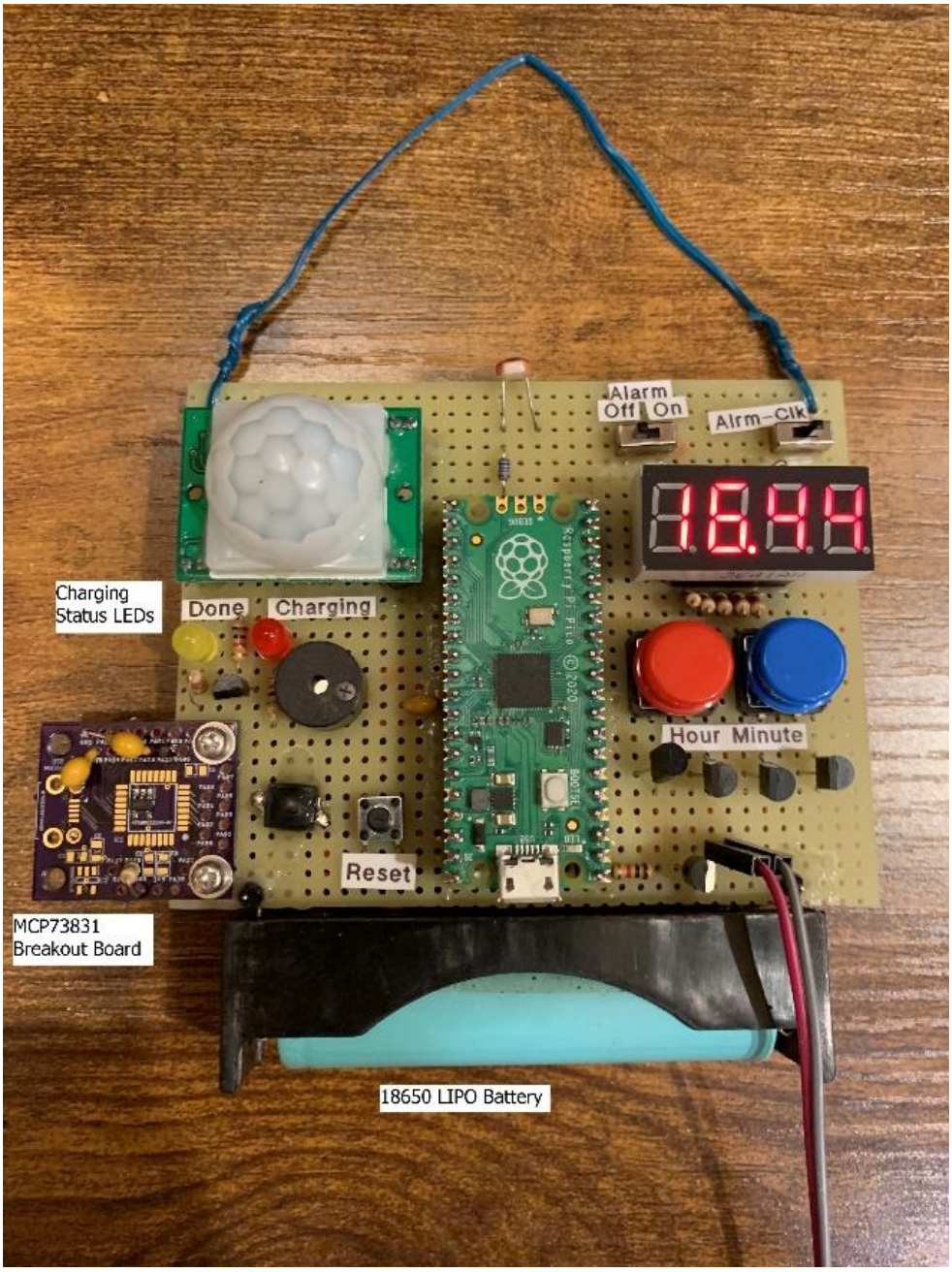




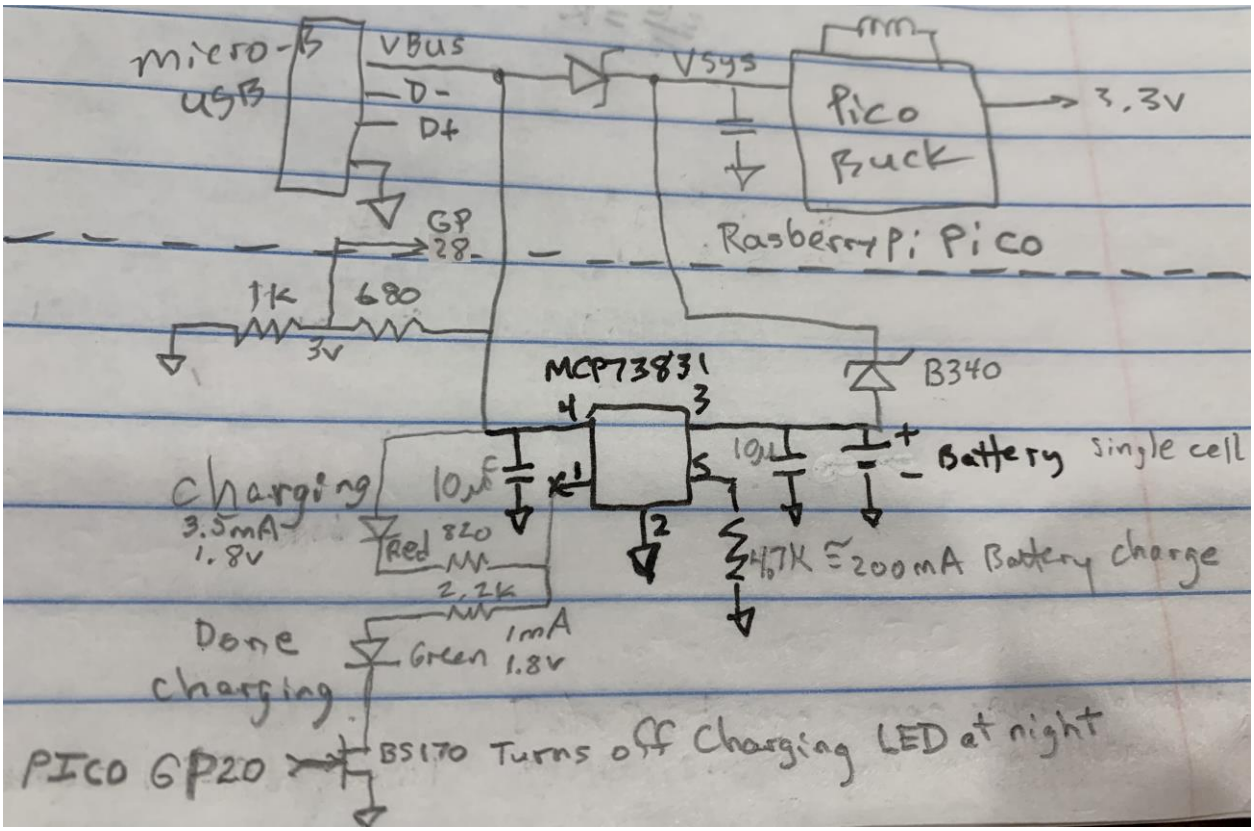




UPDATE - Battery Operation: My neighborhood often has power outages so I added a LIPO battery for backup power. It's mainly so I don't have to re-enter the time on the clock but it also keeps the alarm functioning while I'm asleep. The picture below shows the 18650 LIPO battery attached on the bottom and a MCP73831 charge controller added on the side. I didn't want to solder tiny wires to the MCP73831 legs so I added it's footprint to a board I was getting fab'ed for a different project. This allowed me to easily breakout the pins (it's not pretty). The Adafruit 1904 battery charger would look a lot cleaner.



The schematic below shows how it was wired into the Pico's power path. The MCP73831 data sheet says it has reverse blockage so no input diode is needed. The 4.7KΩ resistor causes the 18650 battery to charge nice and slow at about 200ma. I used a B340 Schottky diode to "or-tie" the battery to the Pico's VSYS pin. A smaller diode could be used but I had an extra B340 on hand. Because I don't want the "Done Charging" LED turned on at night, I use an N channel BS170 FET to tie the LED's cathode to ground. The code sends Pico signal pin GP20 low at night to turn off the FET so the LED stays off. My PIR sensor is a 5 volt device which doesn't work very well in the 3.6 to 4.2 battery voltage range. It detects movement when there isn't any and keeps turning the clock LEDs on and off. A 3.3 volt PIR sensor would solve this problem but for my circuit I added a voltage divider from VBUS to ground. This gives the Pico a 3V signal into GP28 so it knows when the 5 volt power is good. The code stops using the PIR sensor when running from the battery and displays the clock LEDs constantly.



Send me an email at thedalles77@gmail.com if you have any questions. I'll update the code as I gain more Pico and MicroPython experience.