Worksheet 2 - Extending the Whereabout Clock

This worksheet follows on from worksheet 1, however you can use the second template, scanner02.py if you want to start with a clean file.

1 Detecting unknown devices

Now we have our basic scanner we can now start to think about security. It is great that we can show who is at home, but what about displaying if someone you don't know connects to your network?

Hopefully you have a password set on your home network so that only people you know can use it (if you don't please talk to me about this!). Not only could others use your network but they will be able to see / intercept any data you send / receive.

Would you know if someone has discovered (or hacked) your WiFi password or perhaps plugged their own device into the network without you knowledge?

Since we are now scanning our network for devices and a what a "whitelist" of known devices we can easily spot any new devices which may not "belong" on our network.

Adjust the following in the displayDevices() function in your code:

```
def displayDevices(itemseen):
    '''This function allows the devices seen to be displayed'''
    name = "UNKNOWN"
    ident = devices.ident.IGNORE
    mac = itemseen[SCAN.MAC]
    ip = itemseen[SCAN.IP]
    for item in devices.home:
        if item["mac"] in itemseen[SCAN.MAC]:
            name = item["name"]
            ident = item["ident"]
            break
    # Display active devices
    print("#{:<2} {:<17}\t{:15}\t({})".format(ident, mac, ip, name))
    myDisplay.pos(ident, DISP.PS.ON)
    return
```

<-- Your code must match including the break and return!

2a

This time, we set default values for the "ident", "mac", 'lip" and "name" which will be displayed to screen by the last line of code. We also move the print and myDisplay code to the end (outside of the last for...in loop) so that we display devices which weren't in our list of known family devices.

Run you code (again by pressing F5 and save, or via the command line):

```
python3 scanner01.py
```

Press ctrl+c to stop it again.

The output will look similar to the following:

```
SCANNING...
SIMULATION
#-1 6c:72:20:05:54:45   192.168.0.2     (Network NAS)
#-1 40:b4:cd:16:61:16   192.168.0.13    (UNKNOWN)
#-1 84:b5:41:00:88:66   192.168.0.11    (Bob Tablet)
#2  fc:a6:67:44:22:11   192.168.0.10    (Jane Phone)
#2  f8:3d:ff:43:ee:34   192.168.0.5     (Jane Laptop)
```

We can now see that there is an unknown device on our network!

2 Expanding our PiStop display

We can easily add additional PiStops to our display by specifying the additional positions in the hw_id parameter.

PiStop LED numbering

When we add additional an PiStop we can also specify in what order we want to count the LEDs.
The order of the hw_id's will determine the order of the PiStop in the LED numbering.

For example, we can set a PiStop in position A as the first PiStop and add another PiStop in position C as the next one.

```
with DISP.display(hw_id=["A","C"]) as myDisplay:
```

We can also specified the order the LEDs are counted, by using the "vertical" parameter:

```
myDisplay.pos(ident, DISP.PS.ON, vertical=True)
```

```
[2]  [5]
[1]  [4]
[0]  [3]
 |    |
```

```
myDisplay.pos(ident, DISP.PS.ON, vertical=False)
```

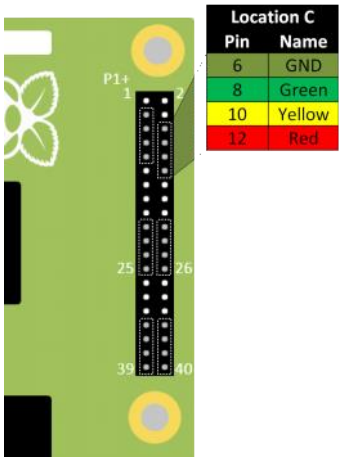
```
[4]  [5]
[2]  [3]
[0]  [1]
 |    |
```

As mentioned on the previous worksheet, if we send a position value less than 0 then we ignore that value. This allows us to set ident.WARN to the last LED (in this case position 5).

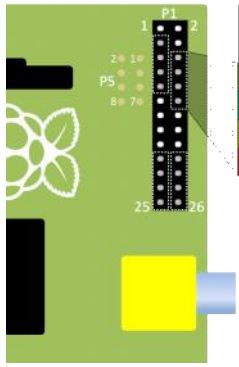
Continued on next page…

3 Displaying an unknown device

Add a PiStop to the Raspberry Pi in position C, as follows:



Location C	
Pin	Name
6	GND
8	Green
10	Yellow
12	Red



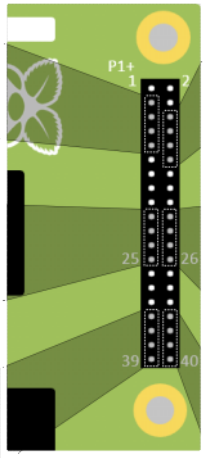
Location C	
Pin	Name
6	GND
8	Green
10	Yellow
12	Red

TIP: *If you want to add more than one PiStop you can do so if they are available.*

Location D	
Name	Pin
Red	3
Yellow	5
Green	7
GND	9

Location B	
Name	Pin
Red	19
Yellow	21
Green	23
GND	25

Location B+	
Name	Pin
Red	33
Yellow	35
Green	37
GND	39



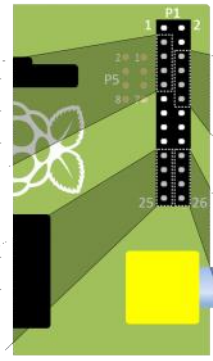
Location C	
Pin	Name
6	GND
8	Green
10	Yellow
12	Red

Location A	
Pin	Name
20	GND
22	Green
24	Yellow
26	Red

Location A+	
Pin	Name
34	GND
36	Green
38	Yellow
40	Red

Location D	
Name	Pin
Red	3
Yellow	5
Green	7
GND	9

Location B	
Name	Pin
Red	19
Yellow	21
Green	23
GND	25



Location C	
Pin	Name
6	GND
8	Green
10	Yellow
12	Red

Location A	
Pin	Name
20	GND
22	Green
24	Yellow
26	Red

The PiStop display is also able to simulate PiStops by displaying them on screen instead (if you use the code on a non-Raspberry Pi computer it will try to simulate it instead - but it will need PyGame installed for it to work. However this is experimental, so currently it is best to display all 6 PiStop positions. You can try this if there aren't any real ones available. This gives you 18 LEDs you can use for the display.

Change the import to use displaysim as follows:

```
import pistop.displaysim as DISP
```

Change the configuration of the myDisplay to initialise all the PiStops (this is the default setting if you don't provide hw_id):

```
with DISP.display() as myDisplay:
```

2b

Change the configuration of the myDisplay by changing your code add position C to the hw_id:

```
with DISP.display(hw_id=["A","C"]) as myDisplay:
```

Change the default ident value to be ident.WARN, so any unknown devices are displayed on the last LED of the display:

```
ident = devices.ident.IGNORE
```

Becomes:

```
ident = devices.ident.WARN
```

Also update the myDisplay.pos command to set vertical to True, since it makes it easier to understand the numbering of the LEDs in this case.

```
myDisplay.pos(ident, DISP.PS.ON, vertical=True)
```

2c

4 Logging unknown devices

While it is helpful to see when an unknown device is connected to your network, this may be something you might notice if you aren't watching at the time. Therefore, we can ensure we record any unknown devices to a file, so we can review them and if required add them to our "known" family device list.

IMPORTANT: You should only perform these commands on networks that you own or have permission to do so. Recording details of devices on a network without permission could be considered a breach of privacy and may otherwise be unlawful. This is particularly important if you are logging this information and keeping it.

First we will add a LOGFILE name and a function for writing the devices to file:

```
SCAN_PERIOD = 10 #seconds
LOGFILE = "unknown.log"

def writeToLog(text, log_file=LOGFILE):
    with open(log_file, 'a') as file:
        file.write("{}\n".format(text))

def displayDevices(itemseen):
```

Next we will update our device checking code to add any unknown devices to our devices list. This ensures that we keep track of previously seen unknown devices (for this run) and just record new ones.

```
def displayDevices(itemseen):
    '''This function allows the devices seen to be displayed'''
    unknown = True
    name = "UNKNOWN"
    ident = devices.ident.WARN
    mac = itemseen[SCAN.MAC]
    ip = itemseen[SCAN.IP]
    for item in devices.home:
        if item["mac"] in itemseen[SCAN.MAC]:
            name = item["name"]
            ident = item["ident"]
            unknown = False
            break
    #Device not known?
    if unknown:
        #Device does not match list (so add)
        unknown_device = {"name"  :"UNKNOWN",
                          "mac"   :mac,
                          "ident" :devices.ident.WARN,
                          "ip"    :ip}
        writeToLog(unknown_device)
    # Display active devices
    print("#{:<2} {:<17}\t{:15}\t({})".format(ident, mac, ip, name))
    myDisplay.pos(ident, DISP.PS.ON, vertical=True)
```

2d

Run you code (again by pressing F5 and save, or via the command line):

```
python3 scanner01.py
```

Press ctrl+c to stop it again.

The result should look like this:

```
SCANNING...
SIMULATION
#-1 6c:72:20:05:54:45  192.168.0.2    (Network NAS)
#5  40:b4:cd:16:61:16  192.168.0.13   (UNKNOWN)
```

unknown.log

```
{'name': 'UNKNOWN', 'ip': '192.168.0.13', 'mac': '40:b4:cd:16:61:16', 'ident': 5}
```

About the code

To write the log file for unknown devices, we create a function that opens a file called "unknown.log". By using the "a" parameter we append (add) to this file each time we write to it, that way any new devices will be recorded.

When we detect an unknown device we set "unknown" to True, allowing us to create a new python dictionary item. By appending this to our devices dictionary the same device won't be detected again as an unknown device, avoiding the same device being logged repeatedly in the log file.

However, if we restart the scanner, then the devices list will be read fresh from the devices.py file and the unknown device will be redetected (and logged again).

Updating the devices list

The output of the log file is such that we can easily cut and paste new devices into our devices file, for example we identified it as another family device.

We can remove warnings of unknown devices by editing the devices.py file and adding a new entry from the unknown.log file.

Edit your devices.py file and add a new line for the detected "UNKNOWN" device. Set the name to something reasonable "John's Kindle" and the ident value to match who it belongs to (or ident.House for devices which stay in the house). You'll have to ensure it keeps to the expected format (i.e. each line except the last must have a comma between it and the next entry in the list).

Re-run your code and you should now find there are no more unknown devices.