



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

This session explores the differences between the

- Microcomputer, or Single Board Computer (SBC) such as Raspberry Pi, and
- Microcontroller such as Arduino

We will then explore a simple Arduino project, Blinking an LED and Wi-Fi Scanning. I will demonstrate an Arduino based NTP clock which I find useful in my shack.

We will discuss possible topics for future sessions and determine what is next.

Contents

BILL OF MATERIALS:	1
DEFINITIONS:	2
HOW TO CHOOSE:	5
EXAMPLE PROJECT THAT USES AN ARDUINO:	6
BUILDING A SIMPLE ARDUINO PROJECT	7
SERIAL MONITOR	12 11
WIFI SCANNING	13 12
WHAT'S NEXT?	15 14
REFERENCES:	19 17
ABBREVIATIONS:	20 19

BILL OF MATERIALS:

- HiLetgo ESP-WROOM-32 ESP32 ESP-32S Development Board, available at Amazon for \$11. You can use a different microcontroller but you will need to adjust your program and wiring.



<< Upload button

<< Reset button

- Computer to host software and power the ESP32
 - Windows, Mac, Linux, including Raspberry Pi, are supported



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

- USB cable to connect them together: MicroUSB to USB-A
- Optional: 10 uF electrolytic capacitor
- From Amazon for \$11: DEYUE 3 Set Standard Jumper Wires Plus 3 Set of Solderless Prototype Breadboard 830 tie Points Breadboard | 3 Set of M/F, M/M, F/F - Each 40pin Electronic Jumpers Wire
<https://www.amazon.com/Standard-Jumper-Solderless-Prototype-Breadboard/dp/B07H7V1X7Y/>
 - Breadboard full size
 - Jumper wires
 - Male-male if all parts are on breadboard
 - Male-female if some parts are not on breadboard
 - Female-female if all parts are not on breadboard
- From Amazon for \$17: Display 320x240 pixel TFT on a carrier board using the ILI9341 driver and an SPI interface. I used "HiLetgo ILI9341 2.8" SPI TFT LCD Display Touch Panel 240X320 with PCB 5V/3.3V STM32"
<https://www.amazon.com/HiLetgo-240X320-Resolution-Display-ILI9341/dp/B073R7BH1B/>
Others are available from sources like EBay and Banggood.

DEFINITIONS:

A microcomputer or SBC (Single Board Computer):

- Runs an Operating System (OS) such as Linux
- An OS provides connections between hardware and your program
 - Internet / networking
 - Disk
 - Keyboard, mouse, monitor
- An OS provides multitasking (multiple programs running concurrently)
 - A single core processor can only do one thing at a time but programs share the processor so they appear to run concurrently.
- An OS provides standard programs
 - Internet browser, media player
 - Many ham radio programs – fldigi, PAT Winlink, Direwolf, code learning, ARDOP, Chirp, APRS, etc.
- An OS provides utilities to support your code (program) or use
 - Examples: backup, configuration control, file viewers

A microcontroller:

- Runs a program without an OS



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

- Provides precise real time control
- Your program must do all the work

How are they the same?

- They can be programmed in C/C++ or Python/MicroPython or other languages
- Great code libraries
- Interrupts supported. Interrupts allow you to interrupt the flow of a program to service a request such as a button press or alarm or clock setting.

How are they different? Note that this is a summary and there could be some more detailed differences:

Feature	Microcomputer-Raspberry Pi	Microcontroller-Arduino
Operating System	Yes	No
Programming	On the Pi	Needs a computer
Language	Any that runs on the OS	C/C++, Python, others with other IDE
Power	Hungry	Efficient
Multitasking	YES	NO
I/O	Digital GPIO	Digital and Analog GPIO Some include internal sensors such as temperature
Disk	MicroSD card provides a file system Supports Hard Drive	Internal memory EEPROM Can support limited external drive but requires library programming
Power control	Shut down before removing power to protect SD card	Just turn off power
Ports	USB, audio, camera, display	May need add on depending on model
Networking	WiFi, Ethernet, Bluetooth	May need add on depending on model
Real time	Not really	Yes
Ham Clock	Another program that can run at the same time as others	Dedicated to this program

Hardware availability and cost:

- Raspberry Pi
 - From UK, not open source
 - Pi 4 Model B 1GB: \$35 plus power, MicroSD, monitor, keyboard and mouse, or VNC
 - Also available: 2GB, 4GB, 8GB



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

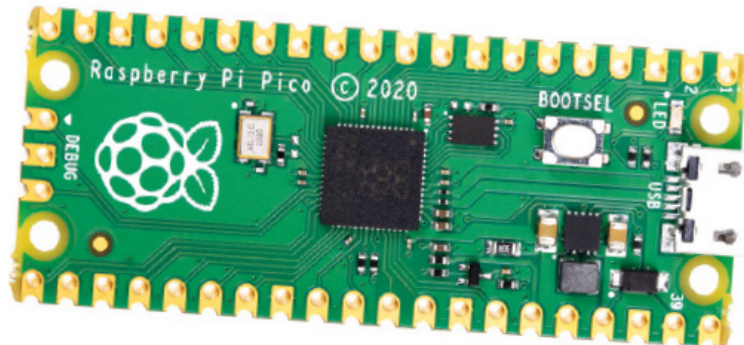
- Pi Zero 2 W: \$15 plus power, MicroSD, etc
- Pi 400, a Pi 4 built into a keyboard \$100 complete
- There are other microcomputers similar to Raspberry Pi
 - Beagleboard
 - Micro:Bit
 - Banana Pi
 - Orange Pi
 - LePotato

- Arduino

- Open source, so there are many variants
 - Any device that can be programmed using the Arduino IDE can be called an “Arduino”
- Original Arduino: Italian (prices include Chinese clones)



- Arduino Uno - \$6 - \$23
- Arduino Nano - \$2 - \$13
- Arduino Pro Micro - \$5 - \$21
- Arduino Mega \$18 - \$40
- Domestic variants from Adafruit, Sparkfun, Teensy
- Chinese variants
 - ESP8266-includes Wifi - \$3 - \$6
 - ESP32-includes WiFi - \$4 - \$17
 - ESP32-CAM-includes ESP32 with onboard camera
- Raspberry Pi PICO, many cool features - \$4-\$6



- Dual core
- Clock is 48MHz vs. usual 16MHz for Arduino



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

- Other companies will use the CPU chip in their products
- You can use a Raspberry Pi or other computer to program and debug a PICO
- MicroPython and C++ support
- Good documentation from Raspberry Pi
- Supported on Arduino
- There is a great book, GET STARTED WITH MICROPYTHON ON RASPBERRY PI PICO, available for free download.
- Includes PIO: Programmable IO to support “bit banging” or implementing communication protocols in really fast software
- PICO W includes WiFi
- See <https://pico.raspberrypi.org/>

HOW TO CHOOSE:

- Determine what you want to do
- See how others have done it
- In some cases, either will work
 - I run remote temperature sensors on Pi Zero AND Arduino
- Raspberry Pi can be used for multiple purposes although you can use it for one purpose
- Arduino is single purpose

Some examples:

What	SBC - Raspberry Pi	Microcontroller - Arduino
Blink an LED	Yes	Yes
Measure resistance	Through a capacitor	Directly through a voltage divider and analog input pin
Measure temperature	Yes	Yes
Winlink	Yes	No
APRS	Yes	Beacon only
YAAC, APRS client	Yes	No, requires Java
WSPR	Yes	Yes
SWR Meter	Yes	Yes
Code practice tutor	Yes	Yes
Oscilloscope	With an Arduino	With a Pi or other display
Multiple Programs	Yes	No, just one
Dedicated use	No, you can develop many programs and select the one you want to use	Yes, dedicated



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

You see that many of these projects could be done with either a Pi or an Arduino. How do you choose?

- If you want to embed your microprocessor/microcomputer in the project, usually an Arduino is the better choice.
- If real time performance is required, usually the Arduino is the better choice.
 - But sometimes, you can fake it with a Pi, such as my Christmas LED display which used a software library that used DMA (direct memory access) to refresh the LEDs while the Pi determines what to display.
- If you need very low power consumption so you can use a battery, usually the Arduino will be the choice, with power management techniques.
- If you want the project to run on a computer, and perhaps use it for several applications, then the Pi is the way to go. For example, Build-a-Pi.

Software libraries to use in building a project are available widely for either choice.

Often, embedding and power requirement and price will drive a selection to use Arduino, while the availability of a program to combine with other functions will push toward a Pi.

EXAMPLE PROJECT THAT USES AN ARDUINO:

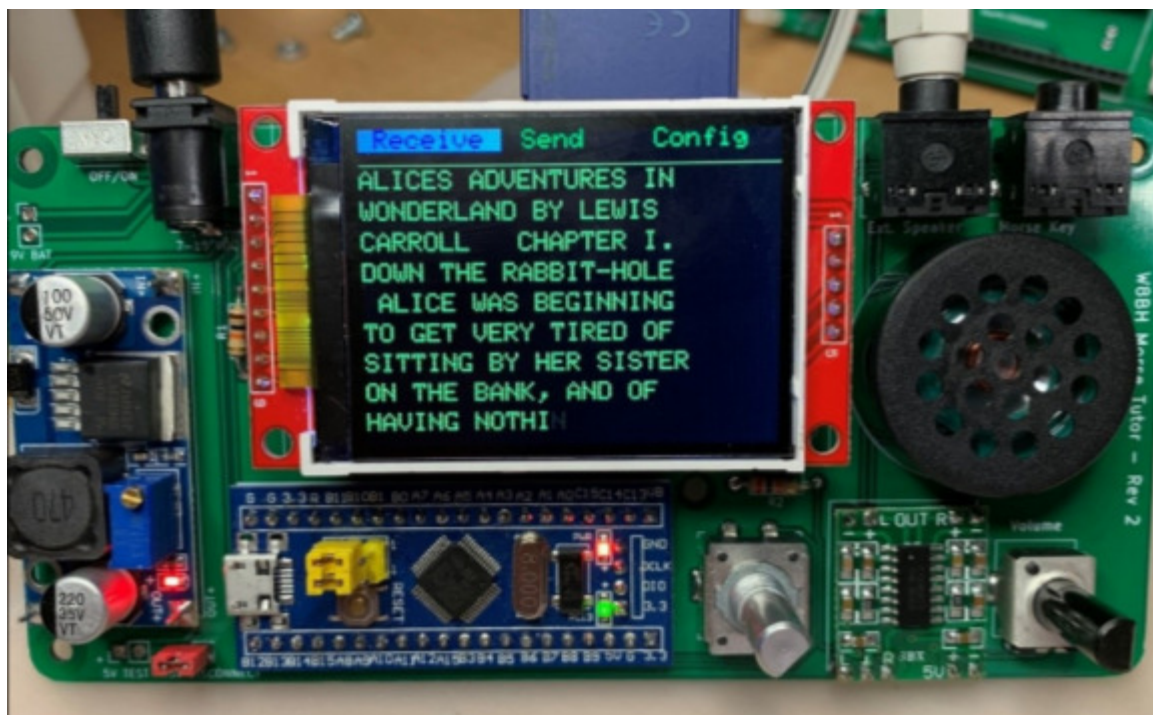
Code practice tutor. W8BH designed one that uses an ESP32 microcontroller called “Blue Pill.” The device is designed to be dedicated to this function.

<http://w8bh.net/MorseTutor1.pdf>

<http://w8bh.net/MorseTutor10.pdf>



1 – PiArduino_V3 - Raspberry Pi vs. Arduino



I built this tutor, although I haven't mastered Morse Code yet.

A good source for Arduino / microprocessor based projects are the books such as:

- Arduino for Ham Radio, by Glen Popiel
- Microcontroller Projects for Amateur Radio, by Jack Purdum
- And many, many online articles, see References at the end of this document.

BUILDING A SIMPLE ARDUINO PROJECT

We will build a simple Arduino project, to **blink an LED**.

To start, we will use the ESP32 microprocessor on a development board. The ESP32-WROOM costs \$11 from Amazon or \$4 plus shipping from Banggood.

Note that a processor with what looks like the same name from a different supplier might be different, and not directly compatible with another. The name, like ESP32, refers to the microprocessor mounted on the development board. The board from a different vendor may have a different pinout (what pins from the processor are made available on the exterior pins of the development board), and what additional circuitry is included on the board.

I recommend that you pick one or maybe two and stick with it/them. I use the Hiletgo ESP32 and the Pico. This is NOT the stable Raspberry Pi environment where every Pi is the same. Buyer beware.



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

For programming, we will start by using the Arduino (free) IDE and the C++ language. C++ is a compiled language so it operates much faster than Python, but your application determines if that is important. It is not important for our first project or for most projects.

You may find this tutorial useful as it shows screenshots of this process:

<https://randomnerdtutorials.com/installing-esp32-arduino-ide-2-0/>

Follow these steps:

1. On your computer, start your internet browser and go to <https://arduino.cc/en/main/software>
There you will find the Arduino software download choices for Windows, macOS, and Linux. Click on the link for Arduino IDE (V2.0.x) that applies to your development environment. For Windows, use "Windows Win 10 and newer."
2. These notes are for the Windows 10 and later environment.
3. You will be asked to support the Arduino project. This is optional. Click either "Just Download" or "Contribute and Download"
4. Click on the file name to install. Follow the usual process and take the defaults.
5. Plug your ESP32 board into your computer using a USB cable. Its red power LED should light.
6. Open the Arduino IDE application that you just installed.
7. Select "File > Preferences"
8. In the Additional Boards Managers URL field, enter https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json on one line. You may need to enter this link manually so it is formatted properly.
9. Select "Tools > Board > Boards Manager..."
10. Scroll down to "esp32". If it is not yet installed, click Install. Wait for download and install to complete then click Close.
11. Select "Tools > Board > ESP32 Arduino > ESP32 Dev Module". This loads the code needed for this particular processor, and is a necessary step for each different type of Arduino.
12. Determine what port your ESP32 is connected to. In Windows, right-click Start, select Device Manager. Select Ports (COM & LPT). Look for Silicon Labs CP210x and note the COM port number. In my case, it was COM4.

NOTE: If you don't see the COM Port in your Arduino IDE, you need to install the CP210x USB to UART Bridge VCP Drivers from:

<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>

Click on Downloads and download the driver appropriate for your operating system environment. For Windows, use the CP210x Universal Windows Driver.



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

After downloading, unzip the zip file and run the appropriate exe, probably

CP210xVCPInstaller_x64.exe

for a 64 bit Windows 10 system.

13. In Arduino IDE, select “Tools > Port > (the COM port number from step 12)”

NOTE: If you don’t see the Port option, your firewall may have blocked the port manager. If so, allow “serial-discovery.exe” to execute.

14. Navigate to <https://randomnerdtutorials.com/installing-esp32-arduino-ide-2-0/>

15. Find “Testing the Installation”

16. Below the code you see, click View Raw Code. A new browser window will open with this code:

```
/*****
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/vs-code-
  platformio-ide-esp32-esp8266-arduino/
  *****/

#include <Arduino.h>

#define LED 2

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED, HIGH);
  Serial.println("LED is on");
  delay(1000);
  digitalWrite(LED, LOW);
  Serial.println("LED is off");
  delay(1000);
}
```

17. Select the code using ctrl-a, then copy the code using ctrl-c


18. In Arduino IDE, click File > New. A new window will open with a skeleton sketch.
Delete the code in this window.

19. Press Ctrl-v and the code will be copied into the window.

20. Press File > Save



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

21. The Arduino IDE will prompt you to name and save the sketch and its directory. Name it “sketch_ESP32blink”
22. In The Arduino IDE, a program is called a “sketch” and has a file type of .ino It is saved in a directory with the same name.
23. In the toolbar, select Upload (the  arrow).
24. The sketch will compile and then try to upload to the ESP32. You can see the progress in the status section at the bottom of the IDE window.
25. When you see “Connecting...” press and hold the button next to the USB port on the opposite side from the red LED, on the right in the above photo of the ESP32, until the IDE shows “Leaving...”.
To eliminate having to press this button, connect a 10 uF capacitor between the EN pin and ground.
26. The blue LED should start blinking on and off once per second. If you don’t see this, press the reset button, the button on the same side as the red LED. Why? Because these are inexpensive devices and there is some variability in them.
27. Power can be removed at any time. When it is restored, the program will run, with the blue LED blinking.
28. You can open this sketch later by selecting “File> Sketchbook > sketch_blinkESP32”

Examine the sketch.

1. Comments begin with “//” or are between pairs of “/*” and “*/” so you don’t have to enter “//” on each line.
2. After the comments, shown starting at /* and ending at */, you will see

```
#define LED 2
```

This statement defines an integer constant with the name of LED and the value of 2. The reason for 2 is that this is the GPIO pin where the built-in LED is attached. If you read the circuit diagram that shipped with your ESP-32S, the LED section shows the wiring of the LED and shows it connected to IO2.

3. Next, you see

```
void setup() {
```

Note there is no ; at the end of this line as it is a function definition and encloses a set of statements that each end with ;

In the Arduino IDE, the “setup()” function is executed once when the sketch first starts.

This block of code ends with

```
}
```



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

4. Next, you see

```
void loop() {
```

All statements within this function are executed repeatedly as long as the sketch runs. When the sketch gets to the } at the end of the block, it goes back to the statement right after:

```
void loop() {
```

5. The setup and loop function definitions have no type, such as integer, so they are defined as a type of void, or no type. We do not need to pass any parameters to them. Some functions expect you to pass a variable to them, which we explain later.
6. In the setup() function, there is a statement

```
pinMode(LED, OUTPUT);
```

This statement sets the mode of the LED pin, which we previously declared as pin 2, to be OUTPUT. We can then set the value of that pin.

pinMode() calls a built-in method to set the mode of the pin. This method expects two parameters, the pin number and the direction, INPUT or OUTPUT.

7. Unlike Python, the indents are purely to make the sketch easier to read, and are not required, but they greatly improve the readability of the sketch.
8. In the loop() function, there is first the statement

```
digitalWrite(LED, HIGH);
```

This statement writes a high voltage, 3.3 volts, to the LED pin so that the LED lights. This is followed by

```
delay(1000);
```

which delays the sketch for 1000 milliseconds, or 1 second, before moving on.

9. Can you see the purpose of the remaining statements?
10. When the sketch reaches the end of the statements within the

```
void loop()
```

function, it returns to the first statement after the

```
void loop()
```



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

and continues through the statements – forever or, more likely, until power is removed or you upload another sketch.

11. This two-function structure is the structure of all Arduino sketches written in the C++ language. Note that C++ allows other structures but the Arduino IDE uses this structure exclusively.

SERIAL MONITOR

Another enhancement is useful to determine what is happening within your sketch. The Arduino IDE provides a connection to your computer so you can interact with your sketch. This is called the Serial Monitor. Let's discuss statements to display what is happening in the sketch to our computer.

We see statements to initialize the Serial Monitor:

```
Serial.begin(115200);
```


This statement begins the serial monitoring at 115200 baud.

```
Serial.println("LED is on");
```

prints one line of text to the serial monitor with a Return at the end of the line. If we wanted to print text without a Return, we would use

```
Serial.print("LED is on");
```



1. Press the serial monitor button  at the top right of the window to start the serial monitor on your computer. Check that the speed is set to 115200 baud.
2. In the Serial Monitor window, you will see the results of the "print" statements in the sketch, or program.

After following the tutorial, you will have:

1. Installed the Arduino IDE
2. Installed support for the ESP32 board
3. Tested the installation by installing and running a program for the Arduino, called a *sketch*.

Let's look at some of the pinout information in the DEVKIT sheet:



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

- GPIO: General Purpose Input/Output pins
- ADC: Analog to digital converter, 12 bits, or values from 0 to 4095
- FLASH: Integrated SPI flash memory
- TOUCH: integrated capacitive touch sensors
- VSPI: SPI bus (A bus is a way to connect devices together)
- TX, RX: asynchronous communications
- I2C: I2C bus

RTC: Deep Sleep state can be used to greatly reduce power consumption. You can wake the processor using timer, touch, or external like pushbutton.

In active state, ESP32 consumes 95-240 milliamps.

In Deep Sleep, it consumes 10-150 microamps. This could be use where you want to wake to take a measurement every 5 minutes, or once a day, for example.

See <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>

WIFI SCANNING

A feature of the ESP32 is that it has WiFi and Bluetooth onboard. As an example of what can be done, I use an ESP32 to remotely monitor the outside temperature on my porch and send that temperature to my weather station running on a Raspberry Pi in my kitchen.

To give you an idea of what the Wifi capability can do, here is a program that will use the ESP32 to scan your area and tell you what WiFi networks are available.



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

sketch_scanWiFi

```
/*
  Example from WiFi > WiFiScan
  Complete details at https://RandomNerdTutorials.com/esp32-useful-wi-fi-functions-arduino/
*/

#include "WiFi.h"

void setup() {
  Serial.begin(9600);

  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  Serial.println("Setup done");
}

void loop() {
  Serial.println("scan start");

  // WiFi.scanNetworks will return the number of networks found
  int n = WiFi.scanNetworks();
  Serial.println("scan done");
  if (n == 0) {
    Serial.println("no networks found");
  } else {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i) {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
      Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN) ? " ":"*");
      delay(10);
    }
    Serial.println("");

    // Wait a bit before scanning again
    delay(5000);
  }
}
```

The comments in the sketch should help you understand how this sketch works.

More info: <https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/>




1 – PiArduino_V3 - Raspberry Pi vs. Arduino

Note that in this sketch, you will see both `Serial.print` and `Serial.println` statements. You will see the result of the difference if you look at the output on the serial monitor and when the Return happens.

To load this sketch, select

File > Examples > (look under the examples for the Examples for ESP32 Dev module) > WiFi > WifiScan

The example WifiScan will load in a new window. Select Upload (the  arrow). The sketch will be compiled and loaded to your ESP32. When the sketch is loaded to the ESP32, it replaces the prior sketch and therefore the LED stops blinking.

Here is what I saw in my Serial Monitor window:

```
scan start
scan done
7 networks found
1: AR300M (-43) *
2: ATT906 (-52) *
3: carol (-53) *
4: Govee_gateway_3C71 (-68)
5: AR300M (-77) *
6: FaryLink_11956B (-78)
7: NETGEAR27 (-93) *
```

Note that this sketch includes a library to provide the WiFi functions. The

```
#include "WiFi.h
```

statement provides this library.

Debugging Note: The first time I tried to upload this sketch to my ESP32, I used a different USB cable. The upload failed. Why? Unknown. When I returned to my original USB cable, it all worked. If you have unexplained things happen, try to determine what has changed, and you may have a hint on how to fix it.

As an exercise, examine other example sketches to learn about what this device can do, and about Arduino C++ programming.

WHAT'S NEXT?

In order for you to experience this workshop and get started on using an ESP32 model of Arduino, get the parts discussed above under Bill of Materials. You can then try this workshop



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

yourself. We will meet for our next session and you can try this, and we will be available to help you get it all working.

Then let's make something useful for your shack.

PROJECT: Ham clock using NTP

W8BH NTP clock using the Hiletgo ESP32 http://w8bh.net/NTP_DualClock.pdf

From W8BH clocks <http://w8bh.net/clocks.pdf>



We will build the circuit on a breadboard so you can get yours working, then if you want to make your clock permanent, we can order circuit boards and the needed parts. We can 3D print cases for your clocks. We will also practice soldering on the circuit boards.

W8BH has other clocks that source the time from other places, including GPS and WWV. These require additional hardware and could be useful in circumstances where you don't have an Internet connection. The time accuracy is the same as for this NTP Time clock.

Summary of the survey:

1. Survey results – Number ranking 4-5 (or yes or maybe) based on 16 responses
 - a. 9 - Day: Saturday
 - b. 4 – Time 9 am
 - c. 16 – Learn about NanoVNA
 - i. NanoVNA is a vector network analyzer for antenna tuning, measuring baluns, cable length and more
 - d. 14 - Raspberry Pi
 - e. 14 – Learn about APRS



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

- f. 14 – Learn about Winlink
 - g. 14 – Build micro projects for fun
 - h. 13 – Build micro projects for ham radio
 - i. 13 - Arduino
 - j. 13 - Transistors, Op amps
 - k. 13 – Learn about programming
 - l. 9 - Basic electronics
 - m. Other topics:
 - i. Satellite tracking systems using telescope drives
 - ii. Back to the basics
 - iii. Software Defined Radio on RPi and Arduino platforms
 - iv. Digital modes of all kinds
2. Problem: Availability of Raspberry Pi
- a. Solution: Start with Arduino
 - b. Raspberry Pi's should be available starting in Q2
3. Date and time: Third Saturday at 9 am / 10 am?
4. Possible sessions
- a. Basic electronics (what? LED, resistor, capacitor, sensor, transistor, op amp?)
 - b. "RF Measurements Using Homemade Equipment," QST, Feb 2023 p.34
series detector probe, shunt detector probe, field strength probe, wavemeter probe, analog using meter and op amp, digital using Arduino Nano
 - c. ESP32 NTP Clock
 - d. ESP32 OTA (Over-the-air) updates: load a new version of a program without needing to plug into the USB port
 - e. Binary clock – show time in binary LEDs
 - f. Hamclock using Arduino and Raspberry Pi
<https://www.clearskyinstitute.com/ham/HamClock/>
 - g. RPi or Arduino WSPR transmitter
 - h. Arduino Morse Code decoder
 - i. Arduino Morse Code tutor
 - j. NanoVNA – Vector Network Analyzer – plots SWR, Smith Chart, Complex Impedances – think antenna and impedance analyzer
 - k. Arduino based antenna analyzer: SWR – built: Antuino
 - l. TinySA – Spectrum Analyzer – plots frequency vs DBm - think waterfall over a broad frequency range
 - m. Raspberry Pi basics – programming and GPIO usage
 - n. Build-a-Pi – will allow you to get up and running fast with a Raspberry Pi for ham radio. Rather than downloading a pre-built image where you have no choices in the build, Build-a- Pi gives you complete control over the build.
 - o. APRS – on Build-a-Pi or Windows or HT or APRSDroid or APRSIS/CE or ?
 - p. Winlink – on Build-a-Pi (PAT) or Windows (Winlink Express)



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

- q. Other Digital Modes:
 - i. FT8 / WSJT-X
 - ii. JS8CALL
 - iii. FLDigi suite
- r. Weather station or home automation platform
- s. Software defined radio:
 - i. Digital Signal Processing theory
 - ii. Use an SDR with Raspberry Pi or Windows
 - iii. ~~T41 Experimenter's Platform: 5 band SSB/CW SDR transceiver~~

WHAT WOULD YOU LIKE TO DO NEXT?



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

REFERENCES:

<http://hiletgo.com/ProductDetail/1906566.html>

<https://circuitdigest.com/microcontroller-projects/programming-esp32-with-arduino-ide>

<https://randomnerdtutorials.com/>

See <https://www.adafruit.com/> for many possibilities of projects and many tutorials. Start with the Learn tab.

Other sources of projects:

<https://create.arduino.cc/projecthub>

<https://www.instructables.com/circuits/arduino/projects/>

<https://www.hackster.io/arduino?ref=topnav>

<https://www.electronicshub.org/arduino-project-ideas/>

<https://all3dp.com/2/arduino-projects-best-sites/>



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

<https://magpi.raspberrypi.com/articles/mini-mars-rover>

uses a Raspberry Pi Pico W (Arduino-like)

<https://magpi.raspberrypi.com/issues/80>

“Amazing Ham Radio Projects” using Raspberry Pi

ADS-B Flight Tracker

WSPR Transmitter

Remote SDR Scanner

Digital Voice Hotspot

Satellite tracking

APRS I-Gate

<https://community.element14.com/learn/publications/ebooks/w/documents/27950/exploring-software-defined-radio-featuring-the-raspberry-pi>

A free E-Book that explores Software Defined Radio and presents some applications using the Raspberry Pi, and links to an image built with SDR applications for the Pi.



1 – PiArduino_V3 - Raspberry Pi vs. Arduino

ABBREVIATIONS:

APRS Automatic Packet Reporting System

DMA Direct memory access

EEPROM Electrically erasable programmable read only memory

GPIO General purpose input output

GPS Global positioning system

IDE Integrated development environment

I2C Inter-integrated circuit

LED Light emitting diode

NTP Network time protocol

OS Operating system

RPi Raspberry Pi

SBC Single Board Computer

SDR Software defined radio

URL Uniform resource locator

USB Universal serial bus

VNC Virtual network computing