Pico W YBox3

Created by Liz Clark

https://learn.adafruit.com/pico-w-ybox3

Last updated on 2024-06-11 12:39:51 PM EDT

Table of Contents

Overview

A little over a decade ago, the YBox2 Kit (http://adafru.it/95) was all the rage. It used a Parallax Propeller chip (https://adafru.it/1a2o) with Ethernet to connect to the Internet and display custom widgets via a composite video output. From ladyada.net (https://adafru.it/1a2p):

> The YBox2 is a DIY networked set-top box. Connect it to your TV and you can design customized content to be delivered direct from the Internet.
>
> YBox2's Memoir
>
> The YBox was first invented by Uncommon Projects (https://adafru.it/1a2q), as part of Yahoo Hack day (https://adafru.it/1a2r). The project was a resounding success, and the following year 80 kits were commissioned by Yahoo for workshops at Maker Faire 2007 (https://adafru.it/1a2s). Robert Quattlebaum (https://adafru.it/1a2t) was one of the lucky few to attend that workshop and decided to see if he could design a second generation YBox (https://adafru.it/1a2u). After much hacking, success! With a little of my help, we were able to design this kit to have more accessories than the original, and just as easy to build, yet less expensive.

Years later, it felt like it was time to revisit this project concept with new hardware. The next generation YBox, the YBox3, uses a Raspberry Pi Pico W to run Arduino code that connects to WiFi and displays custom widgets via DVI output with a DVI PiCowbell.

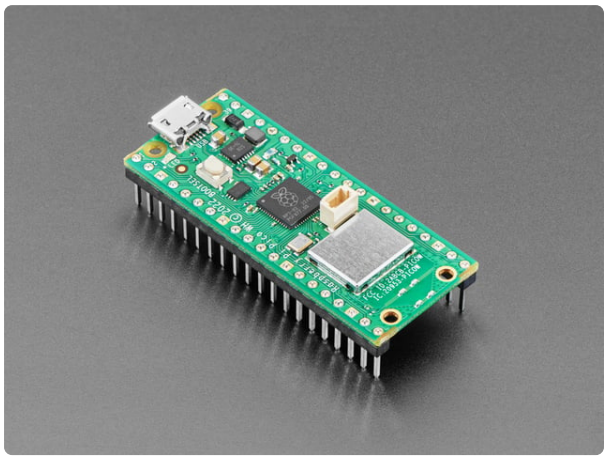
An IR remote changes widget channels on the YBox3 with the help of an IR remote receiver breakout.

Parts

Adafruit PiCowbell DVI Output for Pico - Works with HDMI Display

Ding dong! Hear that? It's the PiCowbell ringing, letting you know that the new Adafruit PiCowbell DVI Output for Pico is in stock and ready...

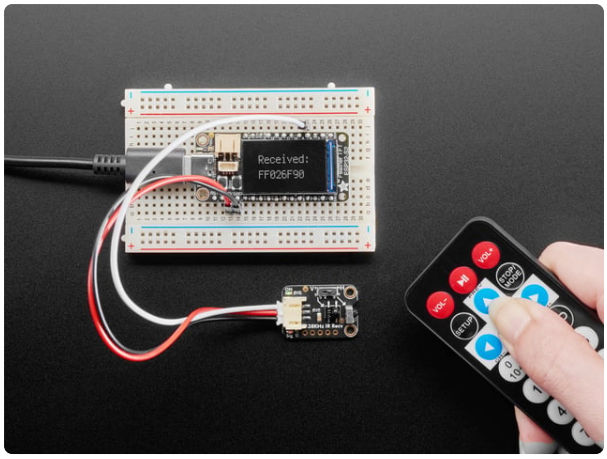
https://www.adafruit.com/product/5745



Raspberry Pi Pico WH - Pico Wireless with Headers Soldered

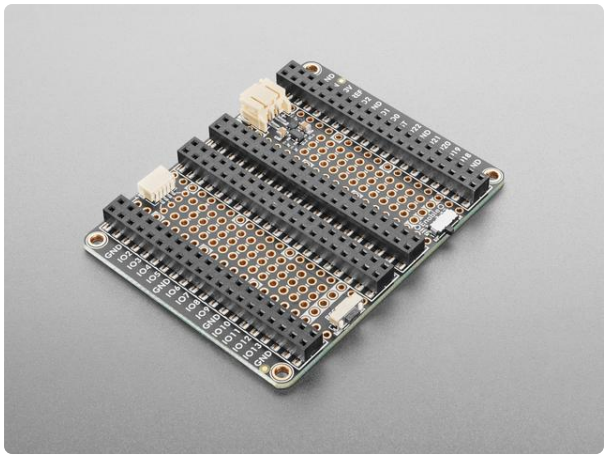
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...

https://www.adafruit.com/product/5544



Adafruit Infrared IR Remote Receiver - STEMMA JST PH 2mm

A year ago we designed a high-current-output Infrared Transmitter
STEMMA which makes it easy to create high-powered...
https://www.adafruit.com/product/5939



Adafruit Proto Doubler PiCowbell for Pico and PicoW

The Adafruit Proto Doubler PiCowBell is intended to be treated like a mini solder-less proto plate to simplify programming and sensor connectivity for your...
https://www.adafruit.com/product/5906

Mini Remote Control

This little remote control would be handy for controlling a robot or other project from across the room. It has 21 buttons and a layout we thought was handy: directional buttons and...
https://www.adafruit.com/product/389

STEMMA JST PH 2mm 3-Pin to Male Header Cable - 200mm

This cable will let you turn a JST PH 3-pin cable port into 3 individual wires with high-quality 0.1" male header plugs on the end. We're carrying these to match up with our...
https://www.adafruit.com/product/3893

5 feet

1 x Mini HDMI to HDMI Cable — https://www.adafruit.com/product/2775

5 feet

---

1 x Micro B USB Cable — https://www.adafruit.com/product/4111

Fully Reversible Pink/Purple USB A to micro B Cable - 1m long

---

1 x HDMI Monitor — https://www.adafruit.com/product/1667

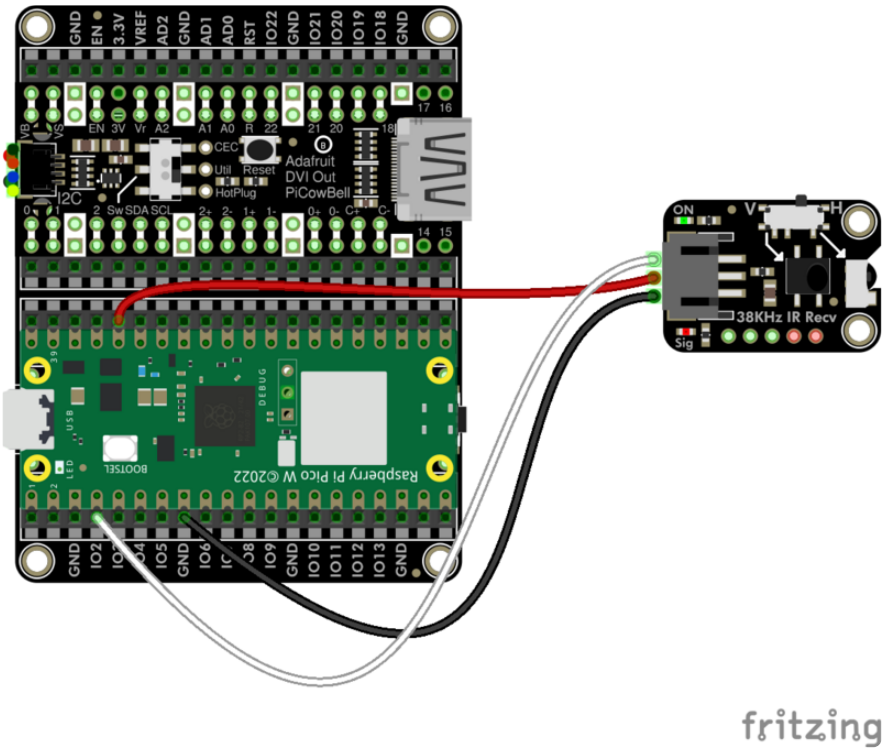
7" Display 1280x800 (720p) IPS + Speakers - HDMI/VGA/ NTSC/PAL

---

1 x M2.5 Screws — https://www.adafruit.com/product/3299

Black Nylon Machine Screw and Stand-off Set – M2.5 Thread

Circuit Diagram



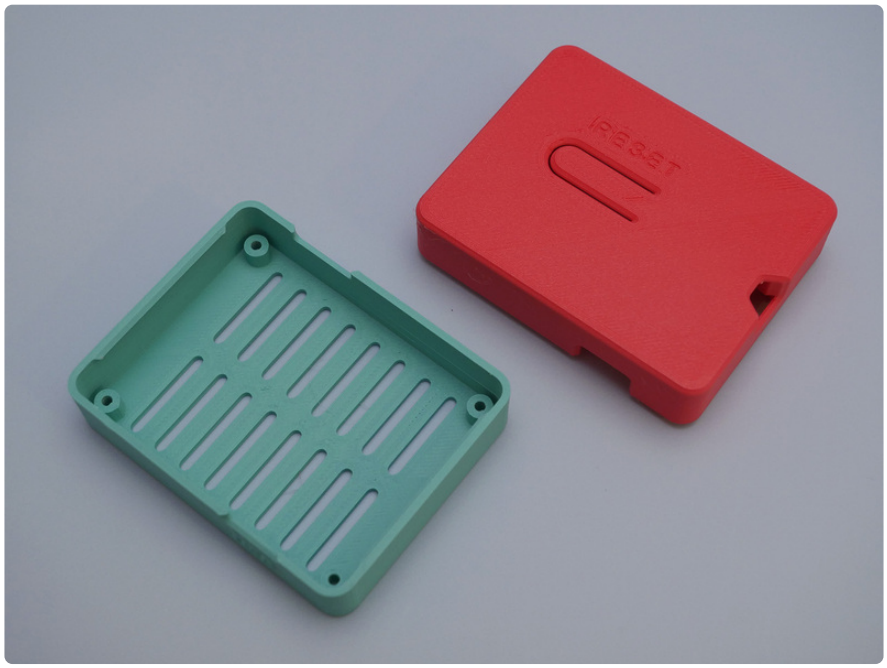
You'll plug a Pico W and DVI PiCowbell into a PiCowbell Doubler.

The IR Remote Receiver breakout plugs into the GPIO headers on the PiCowbell Doubler with a JST PH cable:

- **Breakout Signal** to **Doubler IO2 (white wire)**
- **Breakout VIN** to **Doubler 3.3V (red wire)**

- **Breakout GND** to **Doubler GND (black wire)**

3D Printing

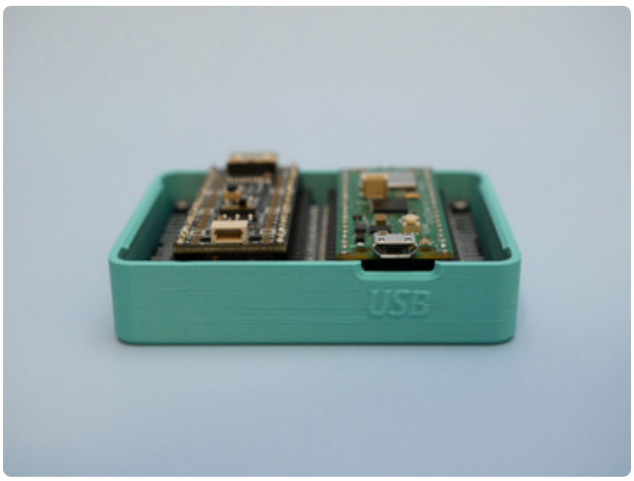


The YBox3 may be assembled with 3D printed parts, described below. The enclosure has two parts: a top lid and a bottom lid.

The STL files can be downloaded directly here or from Printables.

YBox3-3D-Parts.zip

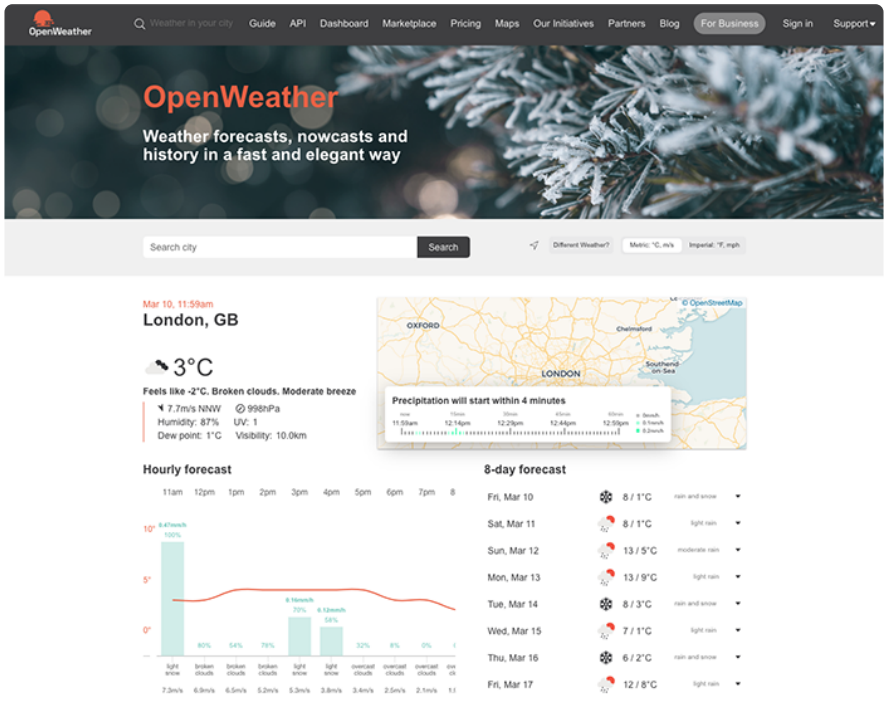
https://adafru.it/1a2v



The case has embossed labels and cutouts for the USB and DVI ports.

The two lids snap fit together. The top lid has a tab that lets you access the reset button on the DVI PiCowbell.

OpenWeather Maps API



Open Weather Maps API Key

We'll be using OpenWeatherMaps (https://adafru.it/KeU) to retrieve the weather info through its API. To do so, you'll need to register for an account and get your API key.

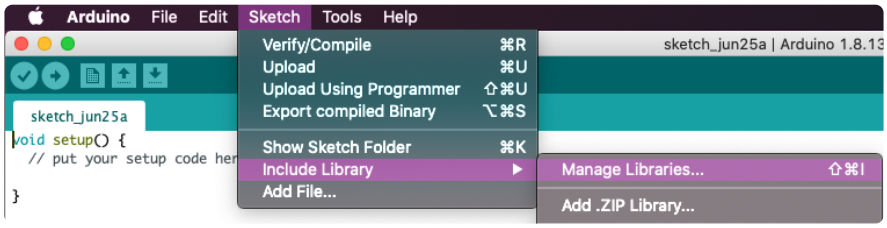
Go to this link (https://adafru.it/EeH) and register for a **free** account. Once registered, you'll get an email containing your API key, known as the "openweather token."

Code the YBox3

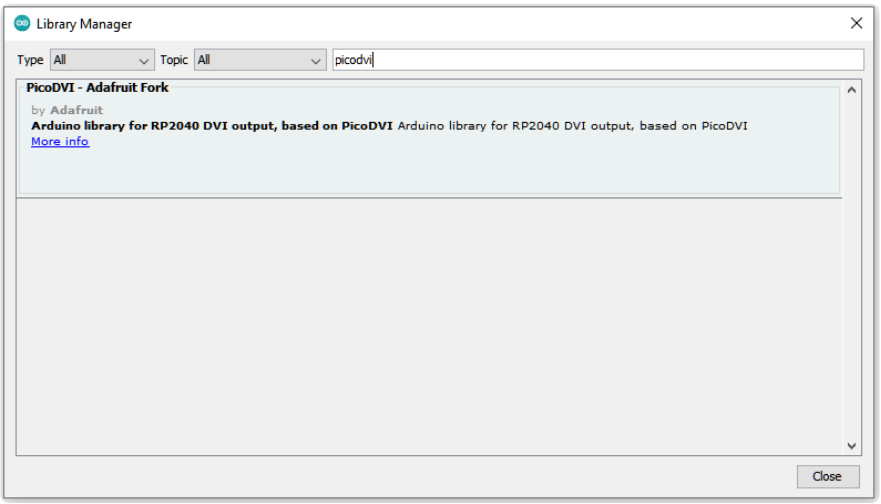
The Pico W has a relatively small amount of memory, and DVI output and WiFi need a lot of it. As a result, this project is coded up using Arduino. You'll need to install the necessary libraries and add your WiFi and OpenWeatherMap credentials before uploading the code to your Pico W with the Arduino IDE.

Install the Libraries

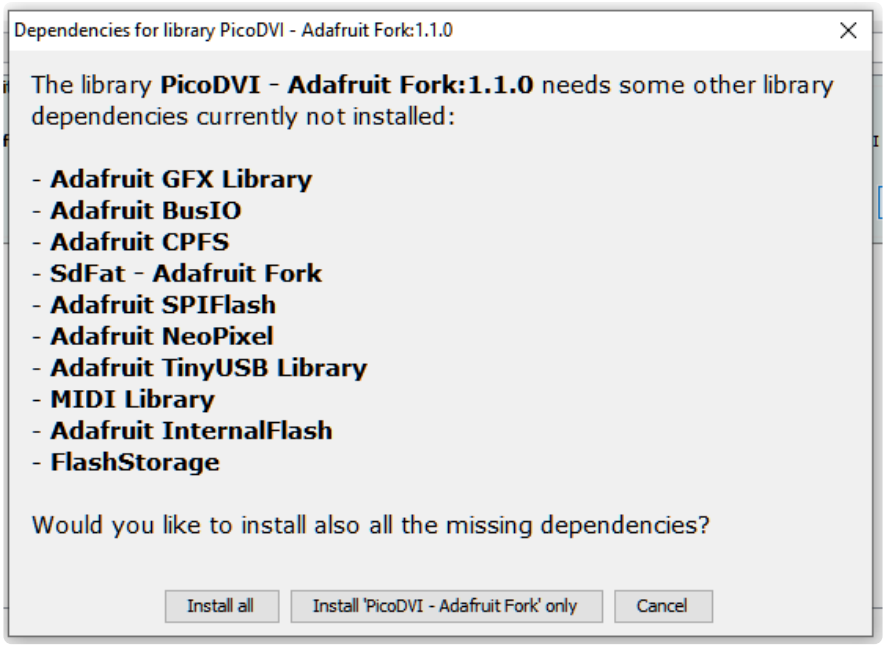
You can install the libraries for this project using the Library Manager in the Arduino IDE.



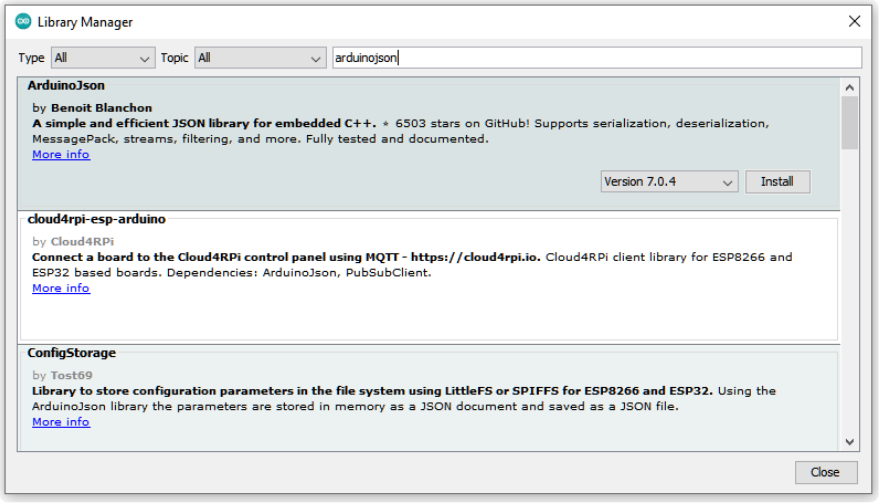
Click the **Manage Libraries...** menu item, search for **Adafruit PicoDVI**, and select the **PicoDVI - Adafruit Fork** library:



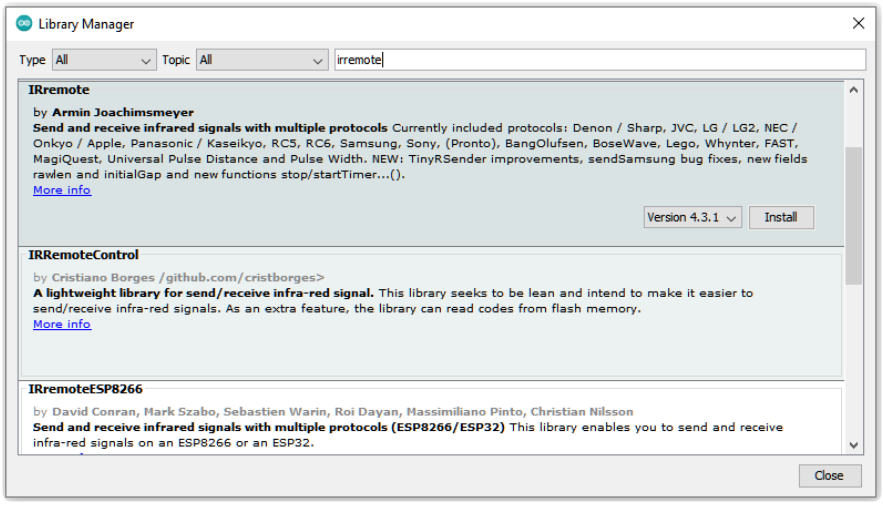
If asked about dependencies, click "Install all".



Then install the ArduinoJSON library. Click the **Manage Libraries...** menu item again, search for **ArduinoJSON**, and select the **ArduinoJSON** library by Benoit Blanchon:



Finally, install the IRremote library. Click on Manage Libraries... again and search for **IRremote** and select the **IRremote** library.



Code Prep

The code consists of a main **.ino** program file and three header files. The header files store the graphics for the project. You'll need all four of these files to properly compile and run the project. These files are available in the .ZIP folder below or on GitHub (https://adafru.it/1a2w).

PicoW_YBox3.zip

https://adafru.it/1a2x

```
// SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// YBox3 with Pico W and DVI PiCowbell
// Bouncing ball screensaver written by Phil B.
// Update user config section with your information

#include <Arduino.h>
#include <time.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <IRremote.hpp>
#include <PicoDVI.h>
#include <Fonts/FreeSansBold18pt7b.h>
#include <Fonts/FreeSans9pt7b.h>

// graphics header files must be included
// in the Arduino IDE project:
#include "graphics.h"
#include "weather_sprites.h"
#include "ybox3.h"

//---------User Config-----------//
#define IR_RECEIVE_PIN   2
const char* ssid = "your-ssid-here";
const char* password = "your-ssid-password-here";
String owm_location = "your-open-weather-maps-location-here";
String owm_key = "your-open-weather-maps-key-here";
// timezone as UTC offset and text
```

```
int timezone = -4;
const char* tz_text = "EST5EDT";
//------------------------------//

int sprite_w = 45;
int sprite_h = 32;
int yPosition = 260;

bool fetch = true;

unsigned long lastUpdate = 0;
unsigned long lastMillis = 0;
unsigned long lastScroll = 0;
int LOOP_DELAY = 30000;
int scrollTime = 30;

char dayBuffer[10];
char dateBuffer[20];
char timeBuffer[10];

String weatherEndpoint = "http://api.openweathermap.org/data/2.5/weather?q=" +
owm_location + "&appid=" + owm_key;
String clockEndpoint = "http://worldtimeapi.org/api/timezone/Etc/UTC";
String apiEndpoint = clockEndpoint;
String channelNow = "clock";

DVIGFX8 display(DVI_RES_320x240p60, true, adafruit_dvibell_cfg);

#define YBOTTOM   123  // Ball Y coord at bottom
#define YBOUNCE  -3.5  // Upward velocity on ball bounce

// Ball coordinates are stored floating-point because screen refresh
// is so quick, whole-pixel movements are just too fast!
float ballx     = 20.0, bally     = YBOTTOM, // Current ball position
      ballvx    =  0.8, ballvy    = YBOUNCE, // Ball velocity
      ballframe = 3;                         // Ball animation frame #
int   balloldx  = ballx, balloldy = bally;   // Prior ball position

void setup() {
  Serial.begin(115200);
  while ( !Serial ) delay(10);

  // Connect to WiFi
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi connected");

  Serial.println("starting picodvi..");
  if (!display.begin()) { // Blink LED if insufficient RAM
    pinMode(LED_BUILTIN, OUTPUT);
    for (;;) digitalWrite(LED_BUILTIN, (millis() / 500) & 1);
  }
  Serial.println("picodvi good to go");
  resetPalette();

  display.swap(true, true); // Duplicate same bg & palette into both buffers
  Serial.println("starting ir..");
  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);
  Serial.println("ir good to go");
```

```
  lastMillis = millis();
}

void loop() {
  if (channelNow == "bounce") {
    // bouncing ball screensaver written by Phil B.
    balloldx = (int16_t)ballx; // Save prior position
    balloldy = (int16_t)bally;
    ballx   += ballvx;         // Update position
    bally   += ballvy;
    ballvy  += 0.06;           // Update Y velocity
    if((ballx <= 15) || (ballx >= display.width() - BALLWIDTH))
      ballvx *= -1;            // Left/right bounce
    if(bally >= YBOTTOM) {     // Hit ground?
      bally  = YBOTTOM;        // Clip and
      ballvy = YBOUNCE;        // bounce up
    }

    // Determine screen area to update.  This is the bounds of the ball's
    // prior and current positions, so the old ball is fully erased and new
    // ball is fully drawn.
    int16_t minx, miny, maxx, maxy, width, height;
    // Determine bounds of prior and new positions
    minx = ballx;
    if(balloldx < minx)                    minx = balloldx;
    miny = bally;
    if(balloldy < miny)                    miny = balloldy;
    maxx = ballx + BALLWIDTH  - 1;
    if((balloldx + BALLWIDTH  - 1) > maxx) maxx = balloldx + BALLWIDTH  - 1;
    maxy = bally + BALLHEIGHT - 1;
    if((balloldy + BALLHEIGHT - 1) > maxy) maxy = balloldy + BALLHEIGHT - 1;

    width  = maxx - minx + 1;
    height = maxy - miny + 1;

    // Ball animation frame # is incremented opposite the ball's X velocity
    ballframe -= ballvx * 0.5;
    if(ballframe < 0)        ballframe += 14; // Constrain from 0 to 13
    else if(ballframe >= 14) ballframe -= 14;

    // Set 7 palette entries to white, 7 to red, based on frame number.
    // This makes the ball spin.
    for(uint8_t i=0; i<14; i++) {
      display.setColor(i + 4, ((((int)ballframe + i) % 14) < 7) ? 0xFFFF : 0xF800);
    }

    // Only the changed rectangle is drawn into the 'renderbuf' array...
    uint8_t  c, *destPtr;
    int16_t  bx  = minx - (int)ballx, // X relative to ball bitmap (can be negative)
             by  = miny - (int)bally, // Y relative to ball bitmap (can be negative)
             bgx = minx,              // X relative to background bitmap (>= 0)
             bgy = miny,              // Y relative to background bitmap (>= 0)
             x, y, bx1, bgx1;         // Loop counters and working vars
    uint8_t  p;                       // 'packed' value of 2 ball pixels
    int8_t bufIdx = 0;

    uint8_t *buf = display.getBuffer(); // -> back buffer

    for(y=0; y<height; y++) { // For each row...
      destPtr = &buf[display.width() * (miny + y) + minx];
      bx1  = bx;  // Need to keep the original bx and bgx values,
      bgx1 = bgx; // so copies of them are made here (and changed in loop below)
      for(x=0; x<width; x++) {
        if((bx1 >= 0) && (bx1 < BALLWIDTH) &&  // Is current pixel row/column
           (by  >= 0) && (by  < BALLHEIGHT)) { // inside the ball bitmap area?
          // Yes, do ball compositing math...
          p = ball[by][bx1 / 2];                // Get packed value (2 pixels)
          c = (bx1 & 1) ? (p & 0xF) : (p >> 4); // Unpack high or low nybble
          if(c == 0) { // Outside ball - just draw grid
```

```
            c = background[bgy][bgx1 / 8] & (0x80 >> (bgx1 & 7)) ? 1 : 0;
          } else if(c > 1) { // In ball area...
            c += 2; // Convert to color index >= 4
          } else { // In shadow area, draw shaded grid...
            c = background[bgy][bgx1 / 8] & (0x80 >> (bgx1 & 7)) ? 3 : 2;
          }
        } else { // Outside ball bitmap, just draw background bitmap...
          c = background[bgy][bgx1 / 8] & (0x80 >> (bgx1 & 7)) ? 1 : 0;
        }
        *destPtr++ = c; // Store pixel color
        bx1++;  // Increment bitmap position counters (X axis)
        bgx1++;
      }
      by++; // Increment bitmap position counters (Y axis)
      bgy++;
  }

  display.swap(true, false); // Show & copy current background buffer to next

  } else {
  if (millis() > (lastUpdate + LOOP_DELAY) or fetch) {
      fetch = false;
      if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin(apiEndpoint);
        int httpResponseCode = http.GET();
        if (httpResponseCode > 0) {
          String payload = http.getString();
          Serial.println(payload);
          DynamicJsonDocument doc(1024);
          DeserializationError error = deserializeJson(doc, payload);
          if (!error) {
            if (channelNow == "weather") {
              String location = doc["name"].as<String>();
              const char* weather = doc["weather"][0]["main"];
              float temperature = doc["main"]["temp"];
              int pressure = doc["main"]["pressure"];
              int humid = doc["main"]["humidity"];
              uint32_t datetime = doc["dt"];
              String icon = doc["weather"][0]["icon"].as<String>();
              String dtBuffer = unixTimeToReadable(datetime);
              float converted_temp = (temperature - 273.15) * 9/5 + 32;
              display.fillScreen(4);
              const uint16_t* bitmap = getIcon(icon);
              display.setCursor(100, 38);
              display.setFont(&FreeSans9pt7b);
              display.setTextColor(8);
              display.print(location);
              display.println(" Weather");
              display.println();
              display.drawRGBBitmap(0, sprite_h / 2, bitmap, sprite_w, sprite_h);
              display.drawRGBBitmap(320 - sprite_w, sprite_h / 2, bitmap, sprite_w,
sprite_h);
              display.setTextColor(10);
              display.println(weather);
              display.setTextColor(7);
              display.print("Temperature: ");
              display.print(converted_temp);
              display.println(" F");
              display.setTextColor(5);
              display.print("Humidity: ");
              display.print(humid);
              display.println(" %");
              display.setTextColor(9);
              display.print("Barometric Pressure: ");
              display.print(pressure);
              display.println(" hPa");
              display.println();
              display.setTextColor(1);
```

```
            display.print("Fetched: ");
            display.println(dtBuffer);
          } else if (channelNow == "clock") {
              String datetime = doc["utc_datetime"].as<String>();
              Serial.println(datetime);
              struct tm timeinfo = parseISO8601(datetime, timezone);
              strftime(dayBuffer, sizeof(dayBuffer), "%A", &timeinfo);
              strftime(dateBuffer, sizeof(dateBuffer), "%B %d, %Y", &timeinfo);
              strftime(timeBuffer, sizeof(timeBuffer), "%I:%M %p", &timeinfo);

          } else {
          Serial.print("deserializeJson() failed: ");
          Serial.println(error.f_str());
        }
      } else {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
      }
      http.end();
    } else {
      Serial.println("WiFi Disconnected");
      WiFi.begin(ssid, password);
    }
   lastUpdate = millis();
 }
}

 if (millis() > (lastScroll + scrollTime)) {
   if (channelNow == "clock") {
     display.fillRect(0, 160, 320, 240-160, 4);
     display.setFont(&FreeSansBold18pt7b);
     display.setTextColor(10);
     display.setCursor(55, yPosition);
     display.println("Today is ");
     display.setCursor(55, yPosition + 40);
     display.setTextColor(7);
     display.println(dayBuffer);
     display.setCursor(55, yPosition + 80);
     display.setTextColor(9);
     display.println(dateBuffer);
     display.setCursor(55, yPosition + 120);
     display.setTextColor(10);
     display.println("The time is ");
     display.setTextColor(5);
     display.setCursor(55, yPosition + 160);
     display.println(timeBuffer);
     display.drawRGBBitmap(0, 0, myBitmapybox3_bmp, 320, 162);

     yPosition--; // Move the text up
     if (yPosition < 0) { // Reset to below the display after text has scrolled up
       yPosition = 260;
     }
   }
   lastScroll = millis();
   display.swap(true, false);
 }
}

 if (IrReceiver.decode()) {
     if (IrReceiver.decodedIRData.protocol == NEC) {
       if (IrReceiver.decodedIRData.command == 0x10) {
           Serial.println("pressed 1");
           display.fillScreen(4);
           fetch = true;
           resetPalette();
           apiEndpoint = clockEndpoint;
           yPosition = 260;
           channelNow = "clock";
           LOOP_DELAY = 30000;
```

```
            display.swap(true, true);
        } else if (IrReceiver.decodedIRData.command == 0x11) {
            Serial.println("pressed 2");
            display.fillScreen(4);
            fetch = true;
            resetPalette();
            apiEndpoint = weatherEndpoint;
            channelNow = "weather";
            LOOP_DELAY = 300000;
            display.swap(true, true);
        } else if (IrReceiver.decodedIRData.command == 0x12) {
            Serial.println("pressed 3");
            channelNow = "bounce";
            // Draw initial framebuffer contents (grid, no shadow):
            display.drawBitmap(0, 0, (uint8_t *)background, 320, 240, 1, 0);
        }
        IrReceiver.resume();
      } else {
        IrReceiver.resume();
      }
    }
}

String unixTimeToReadable(uint32_t unixTime) {
    setenv("TZ", tz_text, 1);
    tzset();

    struct tm timeinfo;
    time_t t = unixTime;
    localtime_r(&t, &timeinfo);

    char dBuffer[30];
    strftime(dBuffer, sizeof(dBuffer), "%m/%d/%y", &timeinfo);

    char tBuffer[10];
    strftime(tBuffer, sizeof(tBuffer), "%I:%M %p", &timeinfo);

    // Return seconds followed by the formatted date and time string
    return String(dBuffer) + " @ " + String(tBuffer);
}

struct tm parseISO8601(String d, int tzo) {
  struct tm tm;
  sscanf(d.c_str(), "%d-%d-%dT%d:%d:%d",
         &tm.tm_year, &tm.tm_mon, &tm.tm_mday,
         &tm.tm_hour, &tm.tm_min, &tm.tm_sec);
  tm.tm_year -= 1900; // Adjust since tm struct expects years since 1900
  tm.tm_mon -= 1;     // Adjust since tm struct expects months from 0-11
  tm.tm_isdst = -1;
  Serial.println(tm.tm_hour);
  Serial.println(tzo);
  tm.tm_hour += tzo; // Apply timezone offset directly
  Serial.println(tm.tm_hour);
  mktime(&tm); // Normalize the tm struct after manual adjustments
  return tm;
}

void resetPalette() {
  display.setColor(0, 0xAD75); // #0 = Background color
  display.setColor(1, 0xA815); // #1 = Grid color
  display.setColor(2, 0x5285); // #2 = Background in shadow
  display.setColor(3, 0x600C); // #3 = Grid in shadow
  display.setColor(4, 0x0000); // black
  display.setColor(5, 0x057D); // blue
  display.setColor(6, 0xB77F); // light blue
  display.setColor(7, 0xE8E4); // red
  display.setColor(8, 0x3DA9); // green
  display.setColor(9, 0xFF80); // yellow
  display.setColor(10, 0xFFFF); // white
```

```
  display.setColor(11, 0x0021);
  display.setColor(12, 0x0020);
  display.setColor(13, 0xeca0);
  display.setColor(14, 0xec80);
  display.setColor(15, 0x0001);
  display.setColor(16, 0xfe43);
  display.setColor(17, 0x31eb);
  display.setColor(18, 0x320b);
  display.setColor(19, 0xad56);
  display.setColor(20, 0x1927);
  display.setColor(21, 0x4a4a);
  display.setColor(22, 0x73ae);
  display.setColor(23, 0x4249);
  display.setColor(24, 0x424a);
  display.setColor(25, 0x4a49);
  display.setColor(26, 0xfbe0);
  display.setColor(27, 0xfc00);
  display.setColor(28, 0xf3e0);
  display.setColor(29, 0x1947);
  display.setColor(30, 0xf400);
  display.setColor(31, 0x0821);
  display.setColor(32, 0x0800);
  display.setColor(33, 0xf500);
  display.setColor(34, 0xfd00);
  display.setColor(35, 0xf520);
  display.setColor(36, 0xfd20);
  display.setColor(37, 0xee43);
  display.setColor(38, 0xf643);
  display.setColor(39, 0xfe23);
  display.setColor(40, 0xee23);
}
```

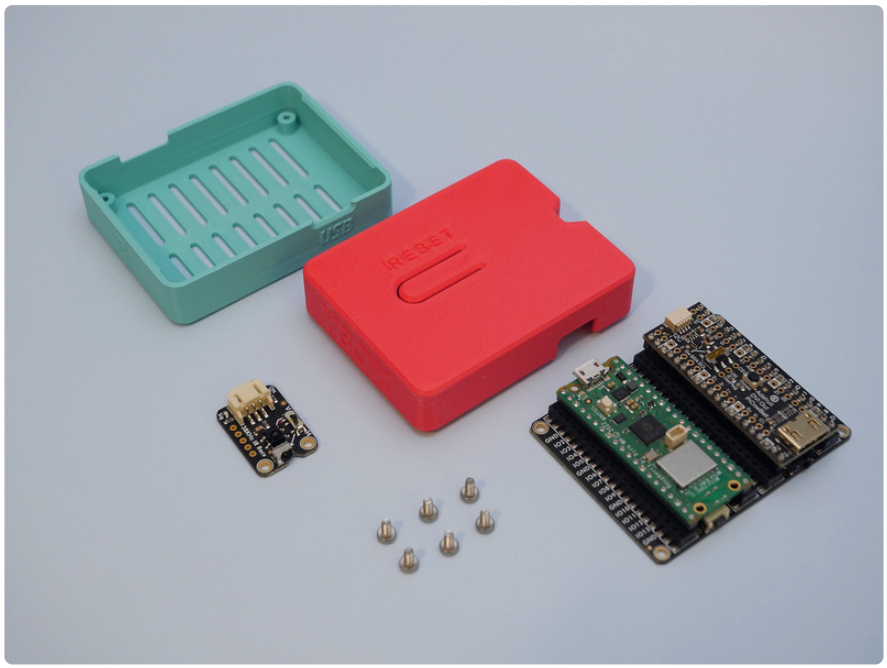
After downloading the files and opening them in the Arduino IDE, navigate to the **.ino** project file. At the top replace the following variables with your connection information:

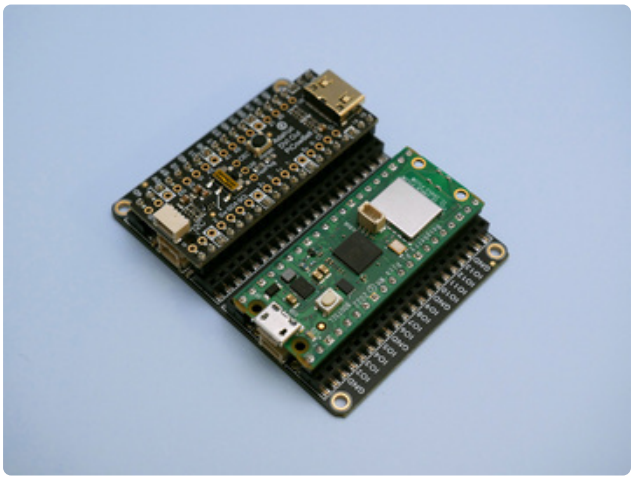
- `IR_RECEIVE_PIN` - the pin for the IR receiver breakout. Defaults to 2.
- `ssid` - your WiFi SSID
- `password` - your WiFi SSID password
- `owm_location` - your location for OpenWeatherMaps, ex: `"Boston,US"`
- `owm_key` - your OpenWeatherMaps key
- `timezone` - your timezone as a UTC offset, ex: `-4` for EST
- `tz_text` - your timezone as text, ex: `"EST5EDT"` for EST

If you don't update the variables at the top of the code, the project will not work!

Upload the sketch to your board. The Pico W will connect to your WiFi network and begin the DVI output. You can use the Serial Monitor in the Arduino IDE for debugging any errors. The code has error messages to let you know if any of the connection variables are missing or not working.

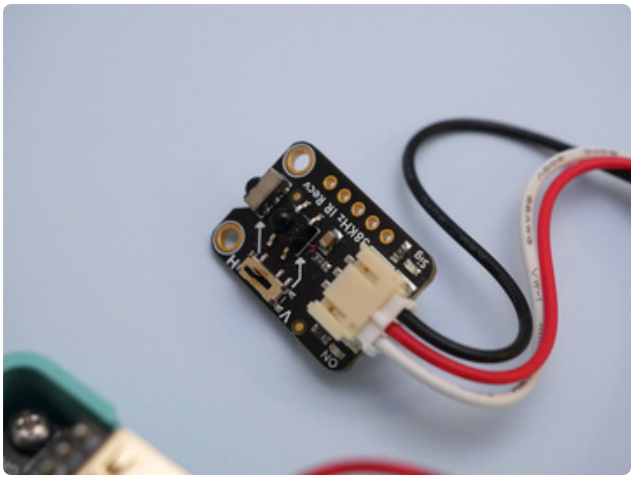
Assembly



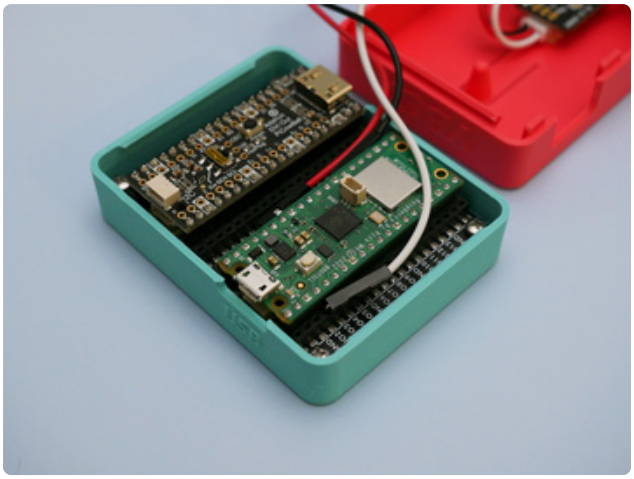


Plug a Pico W and DVI PiCowbell into a PiCowbell Doubler. The Pico W should be plugged into the left side (STEMMA QT/ Reset button) and the DVI PiCowbell should be plugged into the right side (JST PH battery port/slide switch).

Secure the Doubler to the bottom lid of the case with four M2.5 screws.

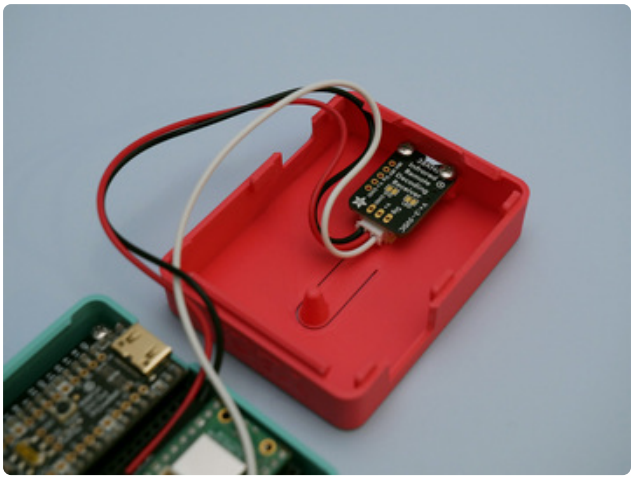


Plug in a JST PH cable with plug headers to the IR breakout. Make sure to **select the horizontal** (**H**) IR receiver with the slide switch.

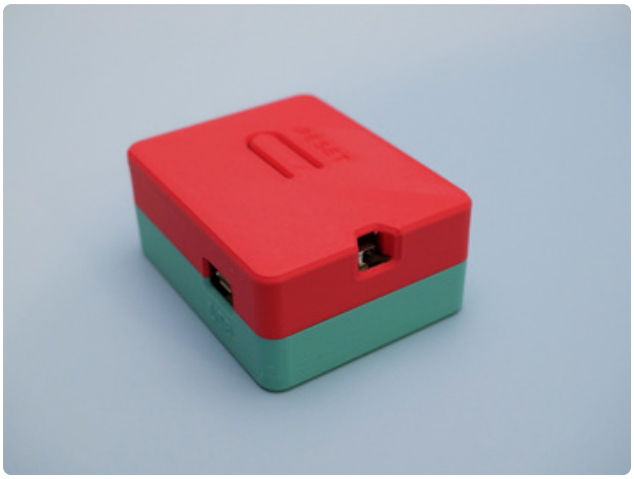


Plug in the JST PH cables to the Doubler:

White wire to **IO2**
Red wire to **3.3V**
Black wire to **GND**
Gently bend the pins down so that the cables will lie flat in the case.



Attach the IR breakout to the case lid with two M2.5 screws. The back of the board should be facing up.



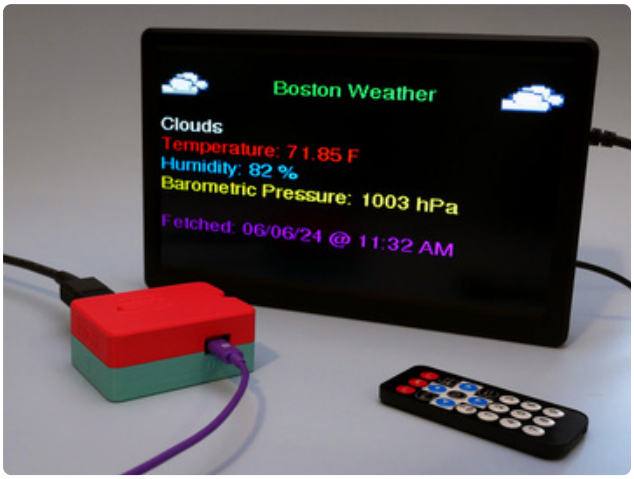
Snap the top and bottom lids together. That completes the assembly!

Use

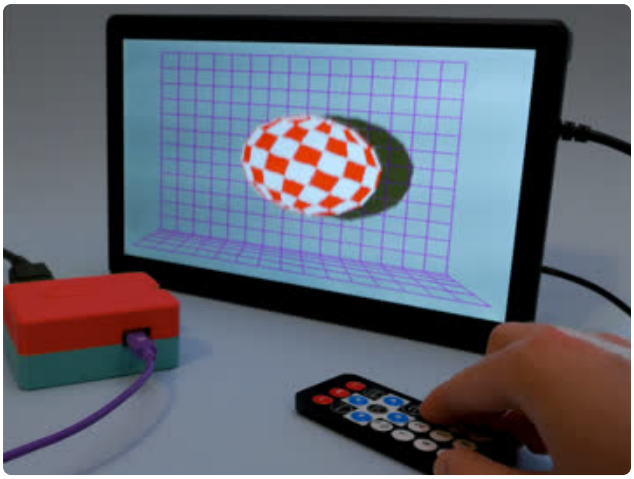
After uploading the code in the Arduino IDE, use a mini HDMI cable or adapter to connect to an HDMI monitor. You'll see the first YBox3 channel display the YBX3 logo and scrolling text showing you the date and time.

You can use the IR remote to change the channels on the YBox3. The **1 button** shows the date/time channel, the **2 button** shows the weather channel and the **3 button** shows the bouncing ball screensaver.

The date and time are fetched from the WorldTimeAPI (https://adafru.it/Of2).



The weather is fetched from OpenWeatherMaps (https://adafru.it/KeU). The code shows the current weather condition, temperature, humidity and barometric pressure. An icon in the upper corners will change depending on the weather condition.



The bouncing ball screensaver (https://adafru.it/1a2y) was coded by Phil B. as a standalone demo for the PicoDVI library.

Going Further

You can customize the YBox3 project code to better fit your needs. You can experiment with different API's, screensavers, colors, etc. If you want to use a different IR remote, you can change the expected IR codes. If you code up your own version, be sure to post it up on Adafruit Playground! (https://adafru.it/18fM)