AtariFruit 2600 Joystick
Created by John Park

https://learn.adafruit.com/atarifruit-2600-joystick

Last updated on 2023-08-29 03:36:06 PM EDT
# Table of Contents

## Overview 3
- Parts & Materials

## Programming the Feather nRF52 5
- iCade Standard
- Feather nRF52 Preparation
- AtariFruit Code

## Joystick Internals 15

## Building the Circuit 20
- Prep the PCB
- Wire Prep
- Solder the Feather Wires
- USB Breakout
- Switch Prep
- Switch to Enable Pin
- 5V to USB
- Shared Grounds
- Feather to PCB Traces
- Battery Power
- Reassembly
- Switch and USB Port
- Charge It

## Playing Games 36
- Bluetooth Pairing
- Play Time
Overview

Playing emulated games on an iPad or iPhone is a lot of fun, but one thing missing from the equation is a proper joystick. To fix this dire situation, you can take an original Atari 2600 joystick and turn it into a wireless controller capable of working with any iCade standard game!

The Feather nRF52 Bluefruit Bluetooth LE microcontroller is perfect for the task. It's fast, has robust Bluetooth, and is tiny enough to fit inside the joystick's case, along with a LiPo battery for power.

You won't even need to modify the case, just remove the original cable and you're ready to create a whole new, modern controller inside of the body of a classic!

You can watch the full build from John Park's Workshop LIVE here:

Parts & Materials

The star of the show is an Atari CX-40 joystick from the venerable Atari 2600 Video Computer System. Check your basement or attic for one, but if you don't luck out, these are available inexpensively -- usually $10 to $15 -- on eBay or at your local thrift shop.

Be sure to get an authentic, vintage, OEM (original equipment manufacturer) Atari joystick made between 1978 to 1986. There are modern joysticks and all-in-one game systems built to look like a CX40, but their internals will vary greatly from the original.
<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Feather nRF52 Bluefruit LE</td>
<td>1 x</td>
<td>nRF52832</td>
<td><a href="https://www.adafruit.com/product/3406">https://www.adafruit.com/product/3406</a></td>
</tr>
<tr>
<td>Lithium Ion Polymer Battery</td>
<td>1 x</td>
<td>3.7v 500mAh</td>
<td><a href="https://www.adafruit.com/product/1578">https://www.adafruit.com/product/1578</a></td>
</tr>
<tr>
<td>USB Micro-B Breakout Board</td>
<td>1 x</td>
<td>extend the USB port</td>
<td><a href="https://www.adafruit.com/product/1833">https://www.adafruit.com/product/1833</a></td>
</tr>
<tr>
<td>Silicone Cover Stranded-Core Wire</td>
<td>1 x</td>
<td>26 AWG black</td>
<td><a href="https://www.adafruit.com/product/1881">https://www.adafruit.com/product/1881</a></td>
</tr>
<tr>
<td>Silicone Cover Stranded-Core Wire</td>
<td>1 x</td>
<td>White</td>
<td><a href="https://www.adafruit.com/product/1882">https://www.adafruit.com/product/1882</a></td>
</tr>
<tr>
<td>Silicone Cover Stranded-Core Wire</td>
<td>1 x</td>
<td>Yellow</td>
<td><a href="https://www.adafruit.com/product/1879">https://www.adafruit.com/product/1879</a></td>
</tr>
<tr>
<td>Silicone Cover Stranded-Core Wire</td>
<td>1 x</td>
<td>Blue</td>
<td><a href="https://www.adafruit.com/product/1878">https://www.adafruit.com/product/1878</a></td>
</tr>
<tr>
<td>Silicone Cover Stranded-Core Wire</td>
<td>1 x</td>
<td>Red</td>
<td><a href="https://www.adafruit.com/product/1877">https://www.adafruit.com/product/1877</a></td>
</tr>
</tbody>
</table>
### Parts Required

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green Silicone Cover Stranded-Core Wire</td>
<td><a href="https://www.adafruit.com/product/1880">https://www.adafruit.com/product/1880</a></td>
</tr>
<tr>
<td>Orange Silicone Cover Stranded-Core Wire</td>
<td><a href="https://www.adafruit.com/product/1883">https://www.adafruit.com/product/1883</a></td>
</tr>
<tr>
<td>5V 1A (1000mA) USB port power supply Charger</td>
<td><a href="https://www.adafruit.com/product/501">https://www.adafruit.com/product/501</a></td>
</tr>
<tr>
<td>USB cable - A/MicroB 3'</td>
<td><a href="https://www.adafruit.com/product/592">https://www.adafruit.com/product/592</a></td>
</tr>
</tbody>
</table>

You'll also need a Philips screwdriver, soldering iron and solder, wire cutters/strippers, and a small bit of double stick foam tape to secure things.

### Programming the Feather nRF52

#### iCade Standard

Since the iPad doesn't have a game controller port on it like a video game console, game developers have come up with a few clever ways to interface physical controls with the device. One such standard is the iCade () protocol, made by ION Audio for their line of iPad desktop arcade cabinets and controllers.
The iCade standard interfaces with iOS as an HID keyboard, paired over Bluetooth Low Energy.

The nRF52 Feather is going to act as a BLE keyboard so that it can "type" the keystrokes that the iCade standard uses.
The press/release scheme allows for button-hold repeat movement and firing, multiple simultaneous commands to work, such as diagonals when an up/down and left/right are hit at the same time, as well as movement and firing at the same time.

Here's what the button mapping looks like for the standard iCade:

The Atari joystick has one single, solitary button, not eight of them like the iCade standard allows! The primary button is mapped to the 'hr' keys, so that's what we'll use!

Feather nRF52 Preparation

If you're new to the Feather nRF52 Bluefruit board, check out this Learning Guide to get started with it.
We'll be programming it as an Arduino board, so make sure you have gone through this setup process () and can successfully upload code to it before proceeding.

For extra credit, you may want to also run some of the example sketches () to make sure you can pair it with your iOS device.

**AtariFruit Code**

In researching the iCade standard for my iPad Pinball Controller project (), I came across an excellent project by Allen C. Huffman () that uses a Teensy as an iCade gamepad controller. We can use a modified version of that code for this project, too!

I made some small adjustments to the code, and included keyboard mapping comments in line, as well as all of the code necessary to convert this to Bluetooth on the nRF52 Feather instead of a hard-wired Teensy.

Copy the code shown below, make a new sketch in Arduino, paste the code into that sketch, save it as AtariFruit_Joystick.ino, and then upload it to your Feather.

Here are the mappings:

- UP on pin 15
- DOWN on pin 7
- LEFT on pin 11
- RIGHT on pin 16
- FIRE on pin 30

```cpp
#include <bluefruit.h>
BLEDis bledis;
BLEHidAdafruit blehid;
```

---

©Adafruit Industries
#define VERSION "0.1"
#define LED_ON

/*
iCade keyboard mappings.

WE     YT UF IM OG
AQ< -->DC
XZ     HR JN KP LV

Atari joystick port, looking at the male DB9 on the Atari.
See: http://old.pinouts.ru/Inputs/JoystickAtari2600_pinout.shtml

1 2 3 4 5/ Up Dn Lt Rt PA
6 7 8 9/ Bt +5 Gd PB
*/

/*
The following I/O pins will be used as digital inputs
for each specific iCade function.
*/
#define UP_PIN    15  // WHT
#define DOWN_PIN  7  // BLU
#define LEFT_PIN  11  // GRN
#define RIGHT_PIN 16  // BRN
#define BTN1_PIN  A2
#define BTN2_PIN  27
#define BTN3_PIN  A0
#define BTN4_PIN  A1
#define BTN5_PIN  30 // ORG
#define BTN6_PIN  A3
#define BTN7_PIN  A4
#define BTN8_PIN  A5

/*
The following keys are the iCade sequence (hold, release)
for each function. Send "W" to indicate UP, and "E" when
UP is released.
*/
#define UP_KEYS    "we"
#define DOWN_KEYS  "xz"
#define LEFT_KEYS  "aq"
#define RIGHT_KEYS "dc"
#define BTN1_KEYS  "yt"
#define BTN2_KEYS  "uf"
#define BTN3_KEYS  "im"
#define BTN4_KEYS  "og"
#define BTN5_KEYS  "hr"
#define BTN6_KEYS  "jn"
#define BTN7_KEYS  "kp"
#define BTN8_KEYS  "lv"
#define DI_PIN_COUNT  12   // 12 pins used.
#define DI_PIN_START  1    // First I/O pin.
#define DI_PIN_END    20   // Last I/O pin.

byte myPins[DI_PIN_COUNT] = {UP_PIN, DOWN_PIN, LEFT_PIN, RIGHT_PIN, BTN1_PIN,
BTN2_PIN, BTN3_PIN, BTN4_PIN,
BTN5_PIN, BTN6_PIN, BTN7_PIN, BTN8_PIN};

char iCadeKeymap[][DI_PIN_COUNT] = {UP_KEYS, DOWN_KEYS, LEFT_KEYS, RIGHT_KEYS,

char iCadeDesc[][DI_PIN_COUNT] = {"Up", "Down", "Left", "Right", "Btn1",
"Btn2", "Btn3", "Btn4",
"Btn5", "Btn6", "Btn7", "Btn8"};

/* We want a very short debounce delay for an arcade controller. */
```c
#define DI_DEBOUNCE_MS 10 // 100ms (1/10th second)
#define LED_PIN 17
//#define POWER_LED 17
#define LEDBLINK_MS 1000

/* For I/O pin status and debounce. */
unsigned int  digitalStatus[DI_PIN_COUNT];          // Last set PIN mode.
unsigned long digitalDebounceTime[DI_PIN_COUNT];    // Debounce time.
//unsigned long digitalCounter[DI_PIN_COUNT];         // Times button pressed.
unsigned int  digitalDebounceRate = DI_DEBOUNCE_MS; // Debounce rate.

/* For the blinking LED (heartbeat). */
unsigned int  ledStatus = LOW;             // Last set LED mode.
unsigned long ledBlinkTime = 0;            // LED blink time.
unsigned int  ledBlinkRate = LEDBLINK_MS;  // LED blink rate.

unsigned int pinsOn = 0;

void setup()
{
    // Just in case it was left on...
    //wdt_disable();

    // Initialize the serial port.
    Serial.begin(115200);
    Serial.println();
    Serial.println("Go to your phone's Bluetooth settings to pair your device");
    Serial.println("then open an application that accepts keyboard input");
    Bluefruit.begin();
    // Set max power. Accepted values are: -40, -30, -20, -16, -12, -8, -4, 0, 4
    Bluefruit.setTxPower(4);
    Bluefruit.setName("AtariFruitJoystick");

    // Configure and Start Device Information Service
    bledis.setManufacturer("Adafruit Industries");
    bledis.setModel("Atari Fruit Joystick 52");
    bledis.begin();

    /* Start BLE HID
    * Note: Apple requires BLE device must have min connection interval >= 20m
    * ( The smaller the connection interval the faster we could send data).
    * However for HID and MIDI device, Apple could accept min connection interval
    * up to 11.25 ms. Therefore BLEHidAdafruit::begin() will try to set the min and
    * max
    * connection interval to 11.25 ms and 15 ms respectively for best performance.
    */
    blehid.begin();

    /* Set connection interval (min, max) to your preferred value.
    * Note: It is already set by BLEHidAdafruit::begin() to 11.25ms - 15ms
    * min = 9*1.25=11.25 ms, max = 12*1.25= 15 ms
    */
    /* Bluefruit.setConnInterval(9, 12); */

    // Docs say this isn't necessary for Uno.
    //while(!Serial) {}

    showHeader();

    // Initialize watchdog timer for 2 seconds.
    //wdt_enable(WDTO_4S);

    // LOW POWER MODE!
```
// Pins default to INPUT mode. To save power, turn them all to OUTPUT
// initially, so only those being used will be turn on. See:
// http://www.pjrc.com/teensy/low_power.html
for (int thisPin=0; thisPin < DI_PIN_COUNT; thisPin++ )
{
    pinMode(thisPin, OUTPUT);
}

// Disable Unused Peripherals
// ADCSRA = 0;

// Initialize the pins and digitalPin array.
for (int thisPin=0; thisPin < DI_PIN_COUNT; thisPin++ )
{
    // Set pin to be digital input using pullup resistor.
    pinMode(myPins[thisPin], INPUT_PULLUP);
    // Set the current initial pin status.
    digitalStatus[thisPin] = HIGH;  //digitalRead(thisPin+DI_PIN_START);
    // Clear debounce time.
    digitalDebounceTime[thisPin] = 0;
    //digitalCounter[thisPin] = 0;
}

// Set LED pin to output, since it has an LED we can use.
pinMode(LED_PIN, OUTPUT);
// pinMode(POWER_LED, OUTPUT);
// digitalWrite(POWER_LED, HIGH);
Serial.println("Ready.");

void startAdv(void)
{
    // Advertising packet
    Bluefruit.Advertising.addFlags(BLE_GAP_ADV_FLAGS_LE_ONLY_GENERAL_DISC_MODE);
    Bluefruit.Advertising.addTxPower();
    Bluefruit.Advertising.addAppearance(BLE_APPEARANCE_HID_KEYBOARD);

    // Include BLE HID service
    Bluefruit.Advertising.addService(blehid);

    // There is enough room for the dev name in the advertising packet
    Bluefruit.Advertising.addName();

    /* Start Advertising
    * - Enable auto advertising if disconnected
    * - Interval:  fast mode = 20 ms, slow mode = 152.5 ms
    * - Timeout for fast mode is 30 seconds
    * - Start(timeout) with timeout = 0 will advertise forever (until connected)
    * *
    * For recommended advertising interval
    * https://developer.apple.com/library/content/qa/qa1931/_index.html
    */
    Bluefruit.Advertising.restartOnDisconnect(true);
    Bluefruit.Advertising setInterval(32, 244);  // in unit of 0.625 ms
    Bluefruit.Advertising.setFastTimeout(30);  // number of seconds in fast mode
    Bluefruit.Advertising.start(0);  // 0 = Don't stop advertising
}

void loop()
{
    // Tell the watchdog timer we are still alive.
    //wdt_reset();

    #ifndef LED_OFF

    ©Adafruit Industries
    Page 11 of 40
// LED blinking heartbeat. Yes, we are alive.
if ( (long)(millis()-ledBlinkTime) >= 0 )
{
// Toggle LED.
    if (ledStatus==LOW) // If LED is LOW...
    {
        ledStatus = HIGH; // ...make it HIGH.
    } else {
        ledStatus = LOW;   // ...else, make it LOW.
    }
// Set LED pin status.
    if (pinsOn==0) digitalWrite(LED_PIN, ledStatus);
// Reset "next time to toggle" time.
    ledBlinkTime = millis()+ledBlinkRate;
}
#endif

// Check for serial data.
if (Serial.available() > 0) {
// If data ready, read a byte.
    int incomingByte = Serial.read();
// Parse the byte we read.
switch(incomingByte)
{
    case '?':
        showStatus();
        break;
    default:
        break;
}
}

// Loop through each Digital Input pin.
for (int thisPin=0; thisPin < DI_PIN_COUNT; thisPin++ )
{
// Read the pin's current status.
    unsigned int status = digitalRead(myPins[thisPin]);

// In pin status has changed from our last toggle...
    if (status != digitalStatus[thisPin])
    {
        blehid.keyRelease();
        // Remember when it changed, starting debounce mode.
        // If not currently in debounce mode,
        if (digitalDebounceTime[thisPin]==0)
        {
            // Set when we can accept this as valid (debounce is considered
            // done if the time gets to this point with the status still the same).
            digitalDebounceTime[thisPin] = millis()+digitalDebounceRate;
        }
        // Check to see if we are in debounce detect mode.
        if (digitalDebounceTime[thisPin]>0)
        {
            // Yes we are. Have we delayed long enough yet?
            if ( (long)(millis()-digitalDebounceTime[thisPin]) >= 0 )
            {
                // Yes, so consider it switched.
                // If pin is Active LOW,
                if (status==LOW)
                {
                    // Emit BUTTON PRESSED string.
                    Serial.print(iCadeDesc[thisPin]);
                    Serial.print(" pressed (sending ");
                    Serial.print(iCadeKeymap[thisPin][0]);
                    Serial.println(" to iCade.");
                    blehid.keyPress(iCadeKeymap[thisPin][0]);
                    //Keyboard.print(iCadeKeymap[thisPin][0]);
                }
            }
        }
    }

```c
//digitalCounter[thisPin]++; pinsOn++;
#ifndef LED_OFF
  digitalWrite(LED_PIN, HIGH);
#endif
} else {
  // Emit BUTTON RELEASED string.
  Serial.print(iCadeDesc[thisPin]);
  Serial.print(" released (sending ");
  Serial.print(iCadeKeymap[thisPin][1]);
  Serial.println(" to iCade.");
  blehid.keyPress(iCadeKeymap[thisPin][1]);
  //Keyboard.print(iCadeKeymap[thisPin][1]);
  if (pinsOn>0) pinsOn--;
  if (pinsOn==0) digitalWrite(LED_PIN, LOW);
}
  // Remember current (last set) status for this pin.
  digitalWrite(thisPin, status);
  // Reset debounce time (disable, not looking any more).
  digitalDebounceTime[thisPin] = 0;
} // End of if ( (long)(millis()-digitalDebounceTime[thisPin]) >= 0 )
#endif
} // End of if (digitalDebounceTime[thisPin]>0)
else // No change? Flag no change.
{
  // If we were debouncing, we are no longer debouncing.
  digitalDebounceTime[thisPin] = 0;
} // End of (int thisPin=0; thisPin < DI_PIN_COUNT; thisPin++ )

// Request CPU to enter low-power mode until an event/interrupt occurs
waitForEvent();

void showHeader()
{
  int i;
  // Emit some startup stuff to the serial port.
  Serial.println("Atari 2600 BlueCade by John Park for Adafruit. ");
  Serial.print("Based on: iCadeTeensy ");
  Serial.print(VERSION);
  Serial.println(" by Allen C. Huffman (alsplace@pobox.com)");
  Serial.print(DI_PIN_COUNT);
  Serial.print(" DI Pins ");
  for (i=0; i<DI_PIN_COUNT; i++)
  {
    Serial.print(myPins[i]);
    Serial.print("=");
    Serial.print(iCadeDesc[i]);
    Serial.print(" ");
  }
  Serial.print(" ");
  Serial.print(digitalDebounceRate);
  Serial.println("ms Debounce.");
}

void showStatus()
{
  showDigitalInputStatus();
}

void showDigitalInputStatus()
{
  Serial.print("DI: ");
  for (int thisPin=0; thisPin < DI_PIN_COUNT; thisPin++)
  {
If you're dying to jump ahead and test out the code before we wire up the joystick, I get it! -- you can try pairing the Feather with your iOS device and grounding each of the assigned pins with a jumper wire. But, don't worry, we'll do more thorough pairing and testing on later in the guide if you're not sure how to go about that.

Once you've uploaded the code, it's time to get inside the joystick and build the AtariFruit circuit!
Joystick Internals


Fig. 3A.

Fig. 3B.

Fig. 4.
The CX-40 is very simple inside, to be sure, and incredibly robust. It consists of a single printed circuit board (PCB) with five contact switches/traces and a ground plane. These are for up, down, left, and right joystick directions, and the fire button.

Normally, these switches are connected directly to the 2600 console by plugging in the cable which is wired to the PCB. Each switch, when pressed, closes a circuit from it's pin to ground.

Let's open it up and take a look inside.

Use a Philips screwdriver to unscrew the four screws from the bottom of the joystick.
Then you can pull open the joystick by carefully prying up the top.

There's a really good chance that 30 years worth of dust and grime has accumulated inside of the joystick. You may want to take this opportunity to wipe that off.

This is the PCB. Each one of those little metal domes is a button, which closes a contact to ground for when pressed.
We won't be using these wires so you can go ahead and take some pliers to pull the crimp connectors off of the board.

Then you can remove the whole cable assembly entirely.
Building the Circuit

Prep the PCB

Instead of crimp terminal connectors, we will solder wired directly to the board. Prepare the board by tinning each pad with a bit of solder.
Wire Prep

Now will prepare to solder connections from the PCB to the Feather microcontroller.
Cut a short length of wire for each contact on the Atari PCB, and one for ground. I've color-coded them similarly to the original wiring. Since I didn't have brown wire I substituted it for yellow.

Strip a bit of insulation off of the ends of each wire.

Solder the Feather Wires

Following this guide, solder each of the direction wires and the fire button wire to the pins of the Feather microcontroller:

- YELLOW: RIGHT to 16
- WHITE: UP to 15
- BLUE: DOWN to 7
- GREEN: LEFT to 11
• ORANGE: FIRE to 30

Since we'll be sharing ground among a few components, wait to solder in the black wire in for now.

USB Breakout
In order to charge the battery we will need to extend the USB port to the outside of the joystick. Prepare the USB breakout board by soldering a red wire to its +5V and a black wire to its ground.
Switch Prep
This slide switch will be used to turn the microcontroller on and off. We can do this by grounding the enable (En) pin on the Feather.

Remove one outside leg from the switch with diagonal cutters, then tin the two remaining leads.
Solder a black wire to each of the switch's leads.
Cover them with heat shrink tubing for insulation.
Use a larger diameter piece of heat shrink to cover both legs.
Switch to Enable Pin
Solder one of the wires from the switch to the enable pin. (It doesn't matter which one.)
5V to USB
Solder the red wire from the USB breakout board to the USB pin on the microcontroller.

This will be used to charge the battery.
Shared Grounds

We will twist together and solder the three ground wires before connecting them to the single ground pin on the Feather. One is from the switch, the second is from the USB breakout, and the third is the wire we'll be connecting to the joystick PCB. This wire can get too thick to fit in the Feather's pin through hole, so be sure to use a light hand with the solder when initially joining the wire ends.

Insert the combined ground wires into the microcontroller's ground pin, then solder the connection.
Feed the slide switch and USB breakout board from the inside of the joystick through the hole and then to the outside.
Feather to PCB Traces
Now place the microcontroller inside of
the joystick body as shown and placed the
original PCB onto its mounting holes
Trim the pin connection wires to a proper
length to fit, with just a bit of extra wiggle
room. If the wires are too long they may
make it difficult to fit everything inside
Next, solder the connecting wires to the
original Atari PCB as shown
Battery Power

Use a little bit of double stick foam tape to secure the battery inside of the joystick body as shown.

Plug the battery’s JST connector into the Feather microcontroller. If the controller turns on, go ahead and flip the switch to turn it off.
Reassembly

Tuck everything in neatly and close the case, being careful not to allow the button spring to fly off!

Secure the joystick with the four original screws.
Switch and USB Port
To secure the slide switch and USB breakout to the joystick’s cable port, use some double stick foam tape.
To maintain the look of the controller, you can blacken the edges of the tape with permanent marker.

Charge It
Now, you’ll be able to plug in the USB cable whenever you need to charge the battery.
Excellent! The joystick is now wireless and ready for play.
Playing Games

Bluetooth Pairing

Before you can play, you'll need to pair the controller with your iPad or iPhone. This is a simple as turning on the joystick and then heading to the Bluetooth settings on your device and choosing the new item listed at the bottom called "AtariFruit Joystick."
You'll be prompted to accept the pairing request. Go ahead and tap "pair".

Now, you'll see the AtariFruitJoystick" has been added to the Bluetooth devices list, with the "Connected" status. It's ready to go!
Once paired, you can test it out by heading to a note-taking app and moving the joystick in all four directions as well as pressing the fire button. You'll see a letter "typed" each time you press or release the joystick from a particular direction, or shoot with the button. That's great, it means everything is working!

Whenever a Bluetooth keyboard is connected, iOS hides the virtual on-screen keyboard. If you need it back, simply turn off the AtariFruit joystick’s switch. The keyboard will reappear. Then, turn the joystick back on to use it again!

Play Time

All that's left is to play! Look for games in the app store that are iCade compatible. The Atari official collection is a great place to start, with it's huge list of 2600 games, all of which use the joystick and single button for control. How about some Adventure?
Another really fun one is Ms. Pac-Man, which, in all honesty should never be played without a joystick!
Enjoy your new/old, wireless classic gaming joystick!