Controlling a Classic Nintendo R.O.B. Robot Using Circuit Playground Express

Created by Anne Barela

https://learn.adafruit.com/controlling-a-classic-nintendo-r-o-b-robot-using-circuit-playground-express

Last updated on 2021-11-15 07:12:37 PM EST

Table of Contents

Overview

After the Video Game Crash of 1983 (https://adafru.it/Bio), console vendors were looking to market new gaming devices without using the word game (seriously). In 1985, Nintendo released their Nintendo Entertainment System (NES) (https://adafru.it/Bip), which proved to be a great hit and the start of a long string of beloved gaming gear.

As part of the first round of shipments, the console came with a light gun (NES Zapper) and a novel interactive robot they called the Robotic Operating Buddy, or R.O.B., which was part of a marketing plan to portray the NES's technology as being novel and sophisticated when compared to previous game consoles, and to portray the NES as being within reach of the better established toy market.

R.O.B. came with some gyro spinners and attachable claws. A lot of the peripherals were soon separated from their robotic friend.

While R.O.B. was appealing, Nintendo only made a couple of games that used his capabilities.

Later NES packages dropped R.O.B., making the price more affordable. R.O.B.s were relegated to closets with many of his little gyro peripherals and his claws scattered and lost.

Revival Efforts

R.O.B. relies on a series of precise flashes from a CRT analog television (https://adafru.it/Biq) to receive commands via a sensor in the robot's head. It had no connection to the NES itself.

Using an NES on a modern LCD TV does not allow R.O.B. to work as the images are not output in the same way analog signals were used. R.O.B. uses a detection chip in his head which relied on specific flash timing inherent in analog TV (in the US, NTSC signals (https://adafru.it/Bir)).

Over the years, people have tried to recreate the R.O.B. controls without success. Some success has been achieved by Makers by hacking the motor control board in R.O.B.'s base (which will be shown later in this article).

Here at Adafruit, we took this as a challenge. With the aid of some NES emulation detective work on the AtariAge forums (https://adafru.it/Bis) along with Ladyada's NTSC-foo, we have recreated the light control sequence R.O.B. uses to move.

This tutorial will assist you in taking your dusty R.O.B. and making him a useful part of your life.

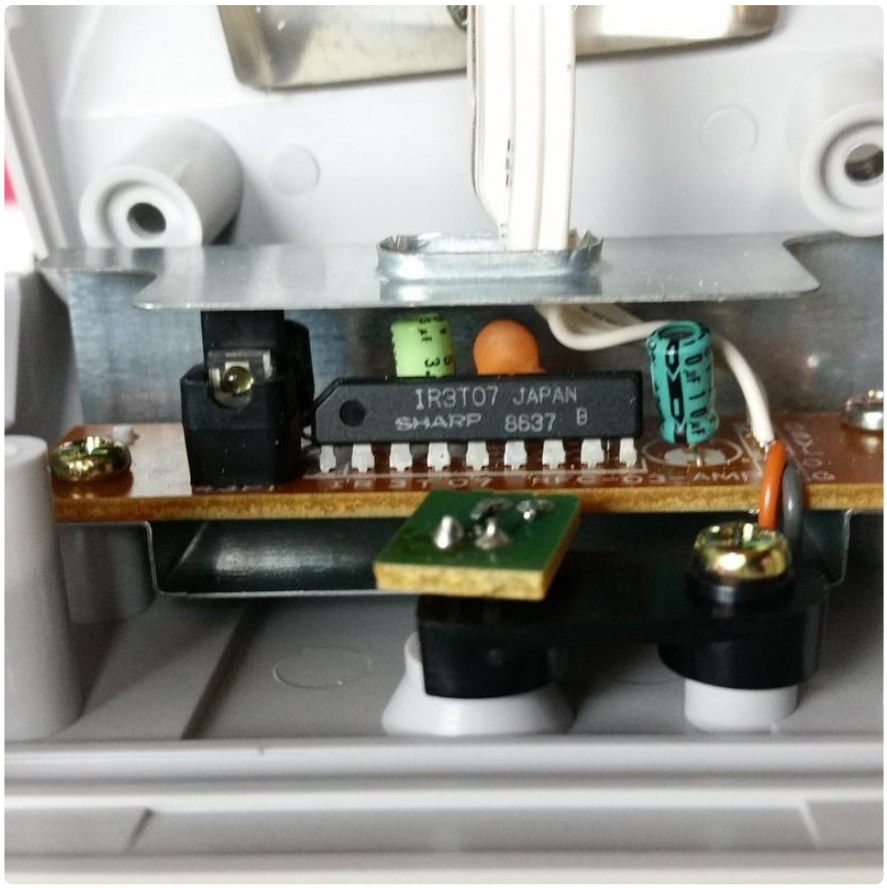
R.O.B. Buying Guide

If you want to go and buy a R.O.B. here are some hints:

1. You don't need an NES console or game cartridge. R.O.B. works alone.
2. You don't need the spinners or other peripherals. The grey "claws" really are not required either.
3. You do want the battery cover for the bottom unless you plan to modify the power in some way.
4. You do want to ask the owner if R.O.B. works with the batteries in and if any gears may be stripped as this will affect the price you pay.
5. Gear issues can often be fixed, but it may be worth it to buy a R.O.B. that seems in working order.
6. You don't need to buy one of the pristine bundles with or without original packaging unless you are a collector and know what you're spending money on. For this tutorial, only the basic R.O.B unit is needed.

As it becomes known that R.O.B. can be controlled, the prices may fluctuate. Before publishing this tutorial, a basic R.O.B. may go for US $50 to $100 on eBay with higher prices for bundles of extra stuff. Do your homework before jumping in. You may have some better luck at thrift shops.

Hacking Optical Control

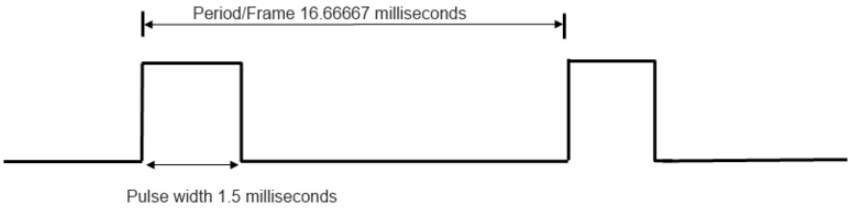


The theory is R.O.B. should be able to be controlled by flashing a light source in the correct manner. R.O.B. uses a phototransistor in his left "eye" connected to a Sharp IR3T07 decoder chip. The IR3T07 is undocumented. But some creative folks at AtariAge.com (https://adafru.it/Bis) reverse engineered old game cartridges to gather R.O.B.s control codes! Tursi (https://adafru.it/Bit) found each command to R.O.B. consisted of 13 bits. The first 5 bits are an initialization string and are always the same: 00010. The next 8 bits are the command, coded as follows:

```
UP:    10111011 - Raises the body up
DOWN:  11111011 - Lowers the body down
LEFT:  10111010 - Turns the body left
RIGHT: 11101010 - Turns the body right
CLOSE: 10111110 - Close the arms
OPEN:  11101110 - Open the arms
TEST:  11101011 - Turn the head LED on
```

In the forum, someone tried to recreate this in Arduino C code but wasn't successful.

Ladyada suggested 60 hz pulses (1/60 = 16.67 milliseconds). No luck. Then she said to use only the blanking time within the 16.67 microseconds which is 1.5 milliseconds. So out of each 60th of a second, have the on/off bit active for 1.5 milliseconds, then off for the remaining 15.167 milliseconds. Also, you must account for some instruction execution time. I shaved the on pulse down time to 15 ms. So the timing is:



This code was tested with an Adafruit 32u4 Feather (Arduino Leonardo compatible) with a white LED. It worked! R.O.B responded to each of the commands!

The LED must be aimed at R.O.B.'s left eye (on the right as you look at him head on). The distance the LED must be to the head varies on what type of LED you are using. Brighter = farther away.

The color of the LED in testing was varied from white to green to infrared. The color did not matter at all! All those green flashes in the NES games - the green color was not the significant factor.

A NeoPixel (https://adafru.it/Bej) smart RGB LED was tried in some early tests. A single NeoPixel could be used but not a ring (apparently a ring takes longer to set and clear the flash of light). A single NeoPixel also seemed to not work as far away as a single color LED and the angle of the light was more critical. So NeoPixels were found not to be ideal for controlling R.O.B.

The Circuit Playground Express can supply a maximum 18 milliamps per external pin with 7 ma recommended. If you decide to use an external LED to control your robot, use a series resistor and chose the value accordingly to avoid excessive current draw which might harm your Circuit Playground Express.

Digikey offers a series resistor calculator for LEDs here (https://adafru.it/Biu).

Control Hardware

Here we use an Adafruit Circuit Playground Express (https://adafru.it/wpF) (CPX) which has the following characteristics:

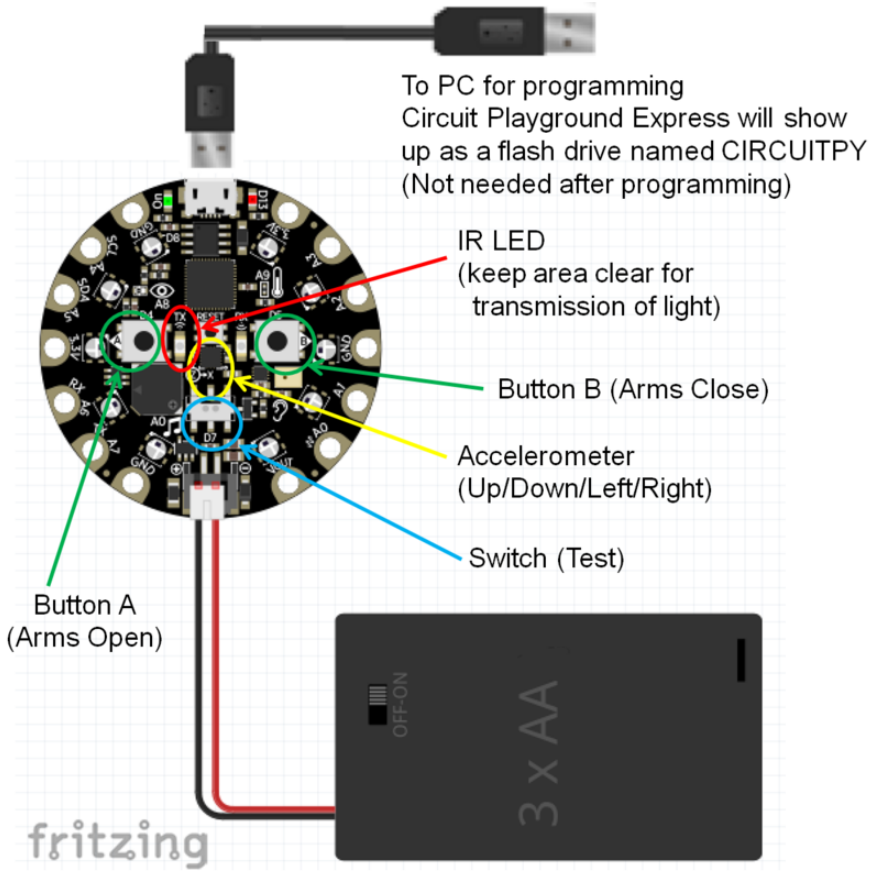
- Has a transmit IR LED with a wide field of view and good transmit power
- CPX shows up on your computer as a USB flash drive when connected via a power+data USB cable
- CPX is programmable via CircuitPython, a language that is easy to use and code that can be changed quickly without installing a toolchain (like Arduino) and changes do not require recompile, just copy a new code file onto CPX and it runs.

Parts List

1 x Circuit Playground Express Programmable in CircuitPython, MakeCode, and Arduino	https://www.adafruit.com/product/3333
1 x Alkaline AA batteries (LR6) Pack of three	https://www.adafruit.com/product/3521
1 x 3 x AA Battery Holder With On/Off Switch, JST connector, and Belt Clip	https://www.adafruit.com/product/3287
1 x USB cable - A/MicroB - 3ft Always use a known good USB cable, not a charge only cable	https://www.adafruit.com/product/592

Build

Hook up the Circuit Playground Express as shown below. When using the USB cable, you don't need the battery pack connected.



For programming and testing, you don't need the battery pack connected, the USB cable will power the Circuit Playground Express and allow for programming.

When programming is complete and you want a stand-alone controller, disconnect the USB cable and plug in the battery pack which has 3 AA cells in it. There is an on/off switch on the battery pack. Note the switch on the Circuit Playground Express is not a power switch.

Programming

The code is a relatively short CircuitPython program. You can learn about CircuitPython at this link to the Adafruit Learning System (https://adafru.it/Biv).

You can copy it from the page or download it from the GitHub link. Save it onto a drive on your local computer as `code.py`.

To load the code as-is, plug your Circuit Playground Board to your computer via the USB cable. You should see a new flash drive called CIRCUITPY in your list of available drives. If you don't see it, press the tiny Reset button on the Circuit Playground Express.

In the off-chance your board does not have CircuitPython preloaded (usually indicated by the board showing up as CPLAYBOOT and not CIRCUITPY), follow the instructions here (https://adafru.it/AFI) to load the latest version. Now press Reset and you should see a drive named CIRCUITPY.

Drag a copy of `code.py` or otherwi copy the file to the CIRCUITPY drive. The program should run immediately.

If you wish to modify and interact with the code, Adafruit suggests using the free Mu editor. You can download Mu and learn more by following this guide (https://adafru.it/Biw). Mu also includes an interactive REPL to show the code is working.

```
# Nintendo R.O.B. control with Accelerometer, code in CircuitPython
# Using an Adafruit Circuit Playground Express board with an IR LED
# Anne Barela for Adafruit Industries, MIT License, May, 2018
# Acknowledgement to info at http://atariage.com/forums/topic/177286
# -any-interest-in-nes-rob-homebrews/ and Limor Ladyada Fried

import time
import gc
from digitalio import DigitalInOut, Direction
from adafruit_circuitplayground.express import cpx
import board

# Commands, each 8 bit command is preceded by the 5 bit Init sequence
Init = [0, 0, 0, 1, 0]            # This must precede any command
Up = [1, 0, 1, 1, 1, 0, 1, 1]     # Move arms/body down
Down = [1, 1, 1, 1, 1, 0, 1, 1]   # Move arms/body up
Left = [1, 0, 1, 1, 1, 0, 1, 0]   # Twist body left
Right = [1, 1, 1, 0, 1, 0, 1, 0]  # Twist body right
Close = [1, 0, 1, 1, 1, 1, 1, 0]  # Close arms
Open = [1, 1, 1, 0, 1, 1, 1, 0]   # Open arms
Test = [1, 1, 1, 0, 1, 0, 1, 1]   # Turns R.O.B. head LED on

print("R.O.B. Start")

# Circuit Playground Express IR LED Setup
IRled = DigitalInOut(board.REMOTEOUT)
IRled.direction = Direction.OUTPUT

# This function performs the LED flashing of a passed command
# Note timing is very tight. Machine instructions are 2-5 ms or so
# so that is why the time for cycles doing work is < 16.66667 ms (1/60)
# Each pulse/space is 1.5 milliseconds out of 16.7 total ms (NTSC)

def IR_Command(cmd):
    gc.collect()                    # collect memory now
    # Output initialization and then command cmd
    for val in Init+cmd:            # For each value in initial+command
        if val:                     # if it's a one, flash the IR LED
            IRled.value = True      # Turn IR LED turn on for 1.5 ms
            time.sleep(0.0015)      # 1.5 ms on
            IRled.value = False     # Turn IR LED off for 15 ms
            time.sleep(0.0150)      # 15 ms
        else:
            time.sleep(0.0167)      # 1 cycle turn off

while True:                         # Main Loop poll switches, do commands
    x, y, z = cpx.acceleration      # Read accelerometer
    print((x, y, z))                # Print for debug
    if x > 5 and y > -8:            # Clockwise from back
```

```
        IR_Command(Right)           # Turn Right
    if x < -5 and y > -8:           # Counterclockwise from back
        IR_Command(Left)            # Turn Left
    if x > 1 and y < -10:           # Move direction of USB
        IR_Command(Up)              # Body up
    if x < -1 and y < -10:          # Move in direction of battery con
        IR_Command(Down)            # Body Down
    if cpx.button_a:
        IR_Command(Open)            # Button A opens arms
    if cpx.button_b:
        IR_Command(Close)           # Button B closes arms
    if cpx.switch:                  # Toggle switch turns "test" on
        IR_Command(Test)            # which is the LED on R.O.B.s head
    time.sleep(0.1)
```

What the code is doing

The code loads libraries for using various functions on the Circuit Playground Express board. Then the definitions for the R.O.B. control bit codes are defined: the `Init` code all commands use and the 8 bit codes unique to various functions: `Up`, `Down`, `Left`, `Right`, `Close`, `Open`, and `Test` (LED).

The function `IR_Command` takes one of the commands above, appends the Init sequence, and turns on and off the IR LED on the Circuit Playground Express according to the timings discussed on page Light Hacking. If the bit is zero, the function delays 1/60 of a second. If the bit is a one, the LED is turned on for 1.5 milliseconds, turned off, and the program waits 15 milliseconds. 15 + 1.5 = 16.5, the other time is used by the code.

The `while True:` loop (exactly like a `loop()` function in Arduino) keeps polling the Circuit Playground Express inputs and outputs commands appropriately.

How each input is used will be explained in the next section.

Use

If you have issues with R.O.B. not responding to the LED commands, carefully open the head up and remove the paper which restricts light to the left eye and blocks the right eye. This will allow more LED light in and make him easier to control.

It is best to experiment close to your computer with the USB cable still connected to the Circuit Playground Express. If you are using Mu, you can click the REPL button and some debug information will show: stating R.O.B. has started and groups of 3 numbers read from the Circuit Playground Express accelerometer (x, y, and z values respectively).

Hold your Circuit Playground Express about 12 to 18 inches from R.O.B.'s head with the IR LED not blocked by your finger and aimed at R.O.B.'s left eye (on your right). Don't shake the Circuit Playground Express yet, hold it steady.

Now press the A button with a finger. R.O.B.'s arms should open wide (if closed). Press button B on the front of the Express and his arms should open. Hey, my robot works!!

The other controls are a bit trickier - some practice will help. If you turn the Circuit Playground Express board clockwise, R.O.B. should twist one way. Turn it counterclockwise and he should move the opposite way. It may take some practice as the the board is reading the changes in acceleration relative to the pull of the earth. Short, quick movements work best while keeping the IR LED aimed at R.O.B.'s head. Tricky but doable.

The final movement involves a quick up movement to have R.O.B. move his torso up. The silver USB connector should be moved up in a short quick movement (without twisting). R.O.B. should move the whole body/arm assembly up. A quick downward movement towards the ground should move the torso down. Again, keep the IR LED pointed at R.O.B.'s left eye. It may take some practice.

Once you have the basics down, you can unplug the USB cable and use the battery pack to make a portable solution. Go have fun with your robot!

Alternative Controls

If you find the movements unsuited for continued use, there are options. You can change the code to increase or decrease the sensitivity of the movements. Use the values output by the Circuit Playground Express to the REPL to guide you on values which may work. Remember that the earth pulls down (in the y direction holding the CPX at R.O.B.) with a value of -9.8 meters/second. Movements add or subtract from that y value.

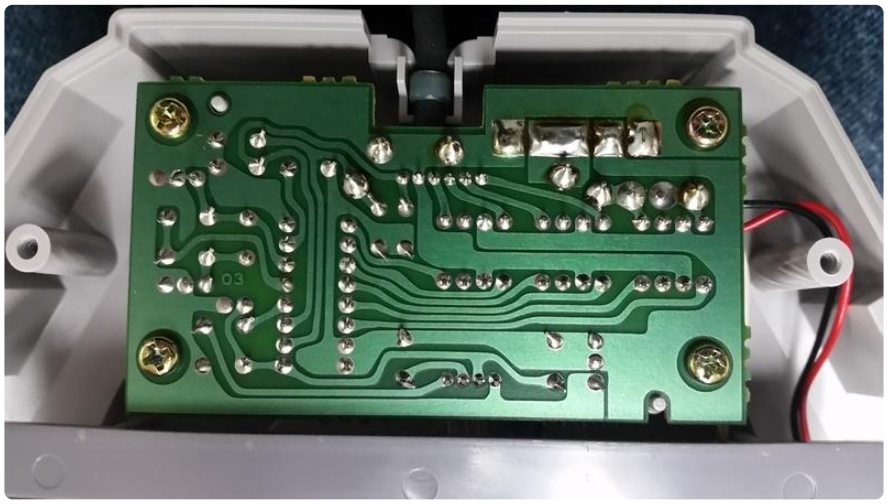
You can add additional push buttons to the Circuit Playground Express pads as a more advanced project. Pads A1 through A7 are capacitive touch switches which can be programmed to activate functions. This is used in the Youtube video on the Overview page.

You can read other buttons to trigger movements. Push buttons should be momentary closed. In that case you might consider some sort of holder/case so you are not juggling your fingers.

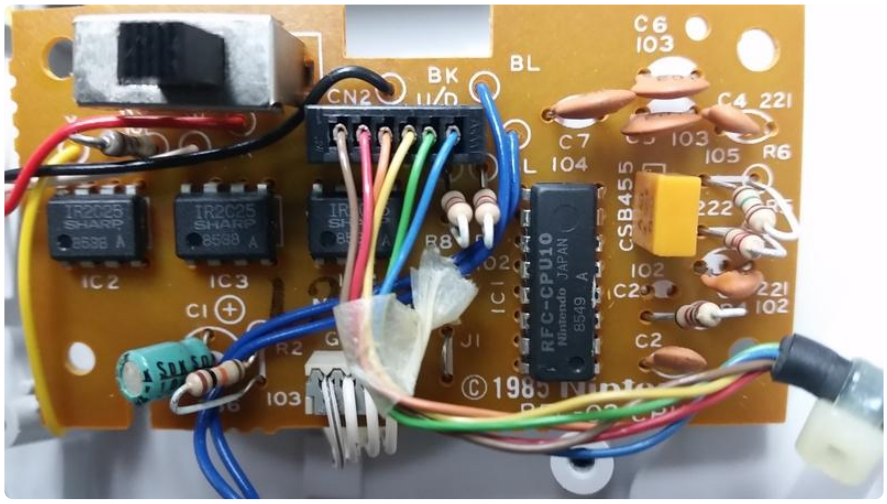
Hardware Hacking

Opening up your R.O.B. could break him. Know the risks. Then again, if you have some hardware skills, there is some interesting things you can do. But you might break things in the process. You have been warned.

The base of R.O.B. contains the other significant circuit board. To access, carefully remove the four screws on the bottom plate. This allows you to see the main circuit board (from the bottom).



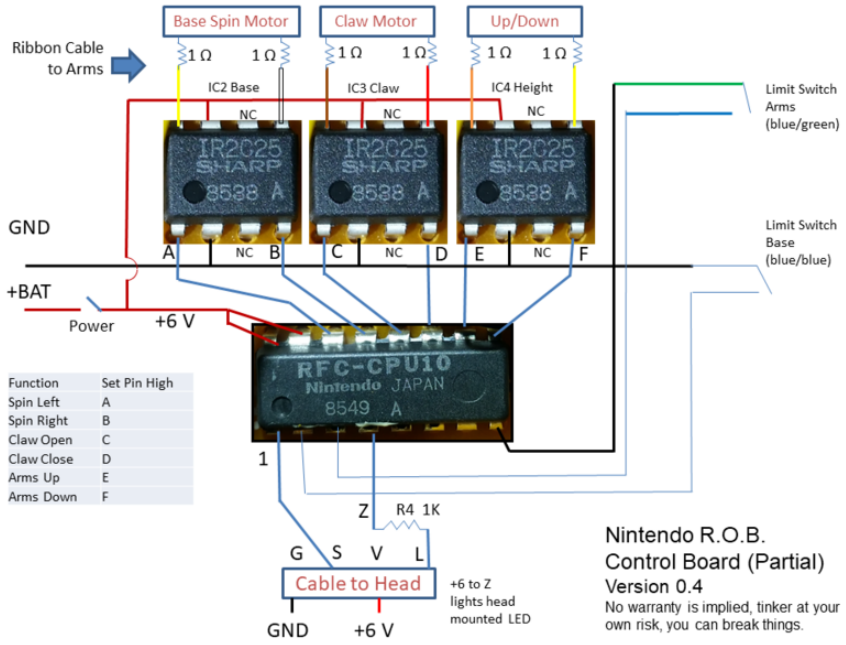
Going further, you can carefully remove the four screws holding on the circuit board to flip it over.



The Sharp IR2C25 chips are undocumented motor control drivers. The RFC-CPU10 is a custom Nintendo microcontroller or programmable chip that is also undocumented. The 6 wire rainbow wiring harness goes to the midsection via a coiled cable. The 4 wire white harness leads to the head section. The slide switch disconnects the battery power from the rest of the circuitry when in the off position.

A partial schematic is shown below. Some of the passive components have been left off, most of them support the CPU chip.

The battery holder uses a plastic tab to hold the base limit switch in a well above the circuit board. Taking the limit switch out risks R.O.B. to excessive body travel which may strip a gear. Yes, I did it and so can you if you are not careful.



Function	Set Pin High
Spin Left	A
Spin Right	B
Claw Open	C
Claw Close	D
Arms Up	E
Arms Down	F

If you have gear issues, like the midbody assembly not moving up & down, a gear has come loose. This can be fixed and is documented in various Youtube and Internet links.

R.O.B. uses three motors to move: one in the base and two in the mid-section. Each motor is controlled by a Sharp IR2C25 driver chip. The CPU takes commands from the white wiring harness wire S connected to pin 1. Then the CPU will connect the appropriate output pin connected to A, B, C, D, E, F, or Z. If you take a wire connected to the positive battery terminal and connect it to one of those connection points noted in the diagram, you will get that function to activate. BUT BE CAREFUL, the limit switches are not communicating to you to stop the movement like the CPU controlling it. If you have a gear issue, you'll have to research the repair - links are here (https://adafru.it/Bix), via Google.

An experienced tinkerer/Maker could use this to build controls to make R.O.B. move. It's not as elegant as using the control signals used earlier and it could damage R.O.B. if you are not careful or you do not take the limitations of movement into account.

Going Further

One of the main purposes Adafruit looked into getting a couple of R.O.B. units was to hack on them. With the control protocol unknown and tube TVs long since sent away, we thought we could take over the motor control.

The new Adafruit Crickit control board and the Circuit Playground Express (https://adafru.it/Biy) could easily help in giving R.O.B. the smarts to bring him to today's standards. Crickit can provide functionality and CPX has all those sensors.

You could use a Crickit along with the existing motor control chips inside R.O.B. This would leave more control for a couple of motors to motorize the base so R.O.B. could move around a room.

If you wanted control besides the existing optical input, you could pair a Wi-Fi or other radio to send commands to Crickit via a receiver (or even use a phone as the controller).

We don't have designs for these or other functions. But there are many guides on the Adafruit Learning System (https://adafru.it/id3) on Crickit, robotics (https://adafru.it/Biz), programming, radios, and much more.