Infrared Hand Gesture Robot Control Glove

Created by Dave Astels

https://learn.adafruit.com/cpx-ir-infrared-hand-gesture-robot-control-glove

Last updated on 2024-03-08 03:06:59 PM EST

Table of Contents

Overview

With a wave of your hand, a robot obeys your command!

This project will take a Circuit Playground Express based robot such as the CRICKIT Snake Bot (https://adafru.it/C1D) or Crickit Carnival Bumper Bot (https://adafru.it/BXS) and add wireless hand gesture control with the built in Infrared.

We'll be using CircuitPython for this project. Are you new to using CircuitPython? No worries, there is a full getting started guide here (https://adafru.it/cpy-welcome).

Adafruit suggests using the Mu editor to edit your code and have an interactive REPL in CircuitPython. You can learn about Mu and its installation in this tutorial (https://adafru.it/ANO).

We'll be making use of the IR transmit and receive capabilities of the Circuit Playground Express to send commands from the controller to the robot. We have a

guide that covers working with Circuit Playground Express IR in detail (https://adafru.it/BX3). There are other project guides that make use of it such as the Treasure Hunt (https://adafru.it/C1E) and Zombie (https://adafru.it/C1F) games.

Parts

You'll need two Circuit Playground Expresses - one on your hand and one for your robot

2 x Circuit Playground Express — https://www.adafruit.com/product/3333
Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. Packed with sensors and programmable with MakeCode, CircuitPython, and Arduino.

---

1 x 500 mAh LiPo Battery — https://www.adafruit.com/product/1578
Or some other small LiPo battery that you can slip into the palm of the glove.

---

1 x LiPo charger — https://www.adafruit.com/product/1905
Keep your robot under control by keeping your controller battery charged.

---

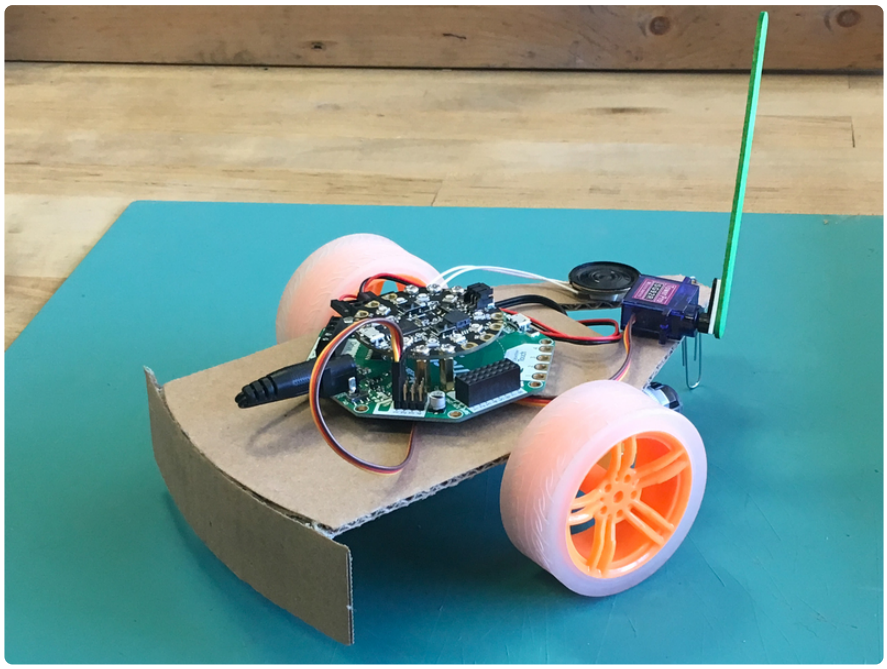
1 x A Circuit Playground Express based robot such as the CRICKIT Snake Bot (https://adafru.it/C1D) or Crickit Carnival Bumper Bot (https://adafru.it/BXS).

Tools and Supplies

A glove onto which you can sew a Circuit Playground Express.

Sewing supplies: needle, thread, scissors. Alternatively you could use self-adhesive Velcro.

Build a Robot



Before you get started with the infrared glove part, we'll need a robot to control.

We recommend building our simple Circuit Playground + Crickit bumper bot. You don't need the speaker and servo flag, the most important parts are the two motors, two wheels, and the Crickit with CPX on top

Visit the Bumper Bot guide to build it

https://adafru.it/BNJ

Motion Control

The only construction required for this guide (beyond a robot) is attaching a Circuit Playground Express to the palm of a glove. A biking glove works well, as you can see in the photos. The alligator clip pads are ideal for sewing; use a few to attach the board to the glove. Regular thread is all that is required, we're sewing it to attach it, not to make any connections.

Attach the Circuit Playground Express so that the battery connector is closest to your wrist, as shown in the photo below. Not only does this align with the accelerometer data expected by the code, it also makes it easy to slip an attached LiPo battery into the glove.

The controller recognizes 9 different hand positions/commands:

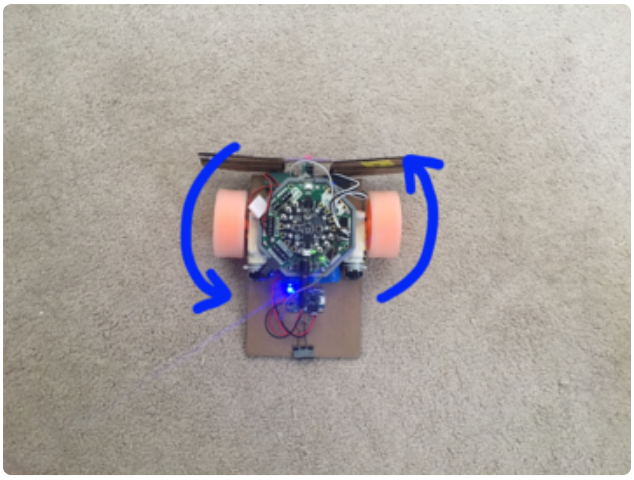
STOP

Hold your hand flat with your palm facing down.

The robot stops all motion.

ROTATE_LEFT

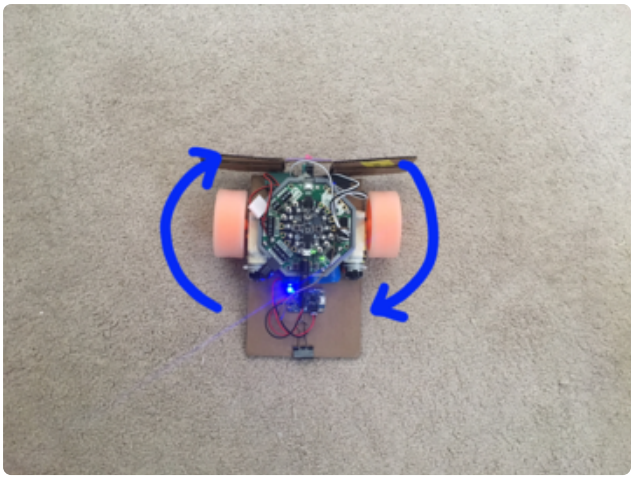
With you hand flat, palm facing down, rotate your wrist counterclockwise.



The robot rotates counterclockwise.

ROTATE_RIGHT

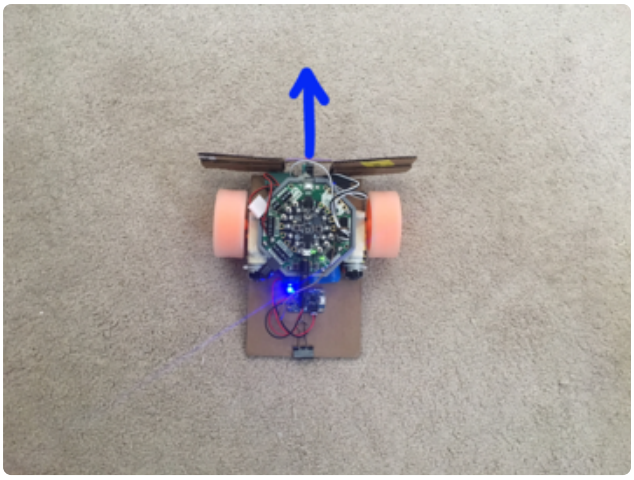
With you hand flat, palm facing down, rotate your wrist clockwise.



The robot rotates clockwise.

FORWARD

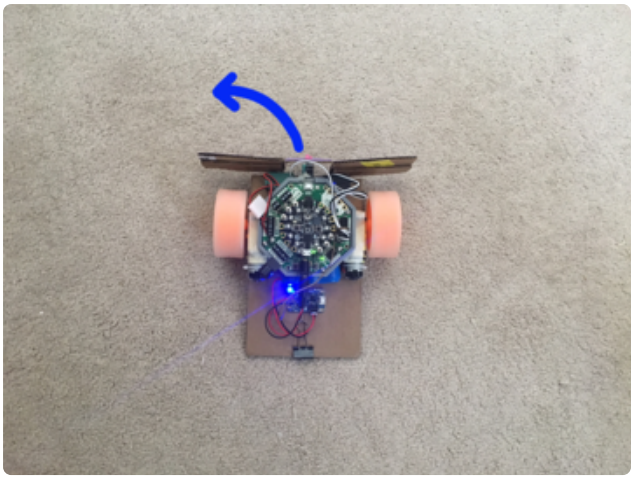
Hold your hand with your palm facing away from you.



The robot will move forward.

FORWARD_LEFT

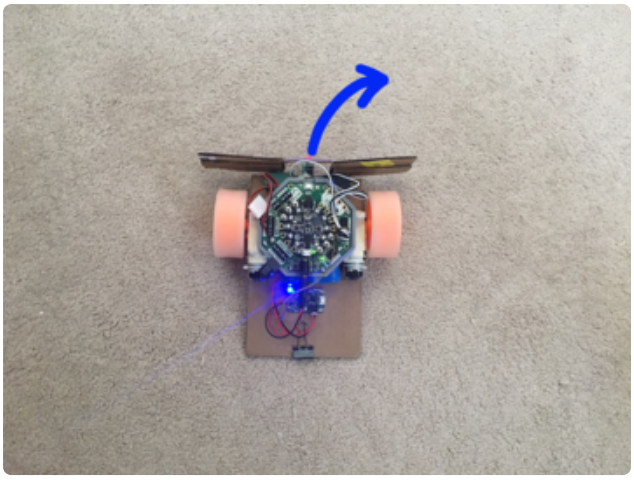
With your hand held palm outward, rotate your wrist counterclockwise.



The robot will veer to the left while moving forward.

FORWARD_RIGHT

With your hand held palm outward, rotate your wrist clockwise.

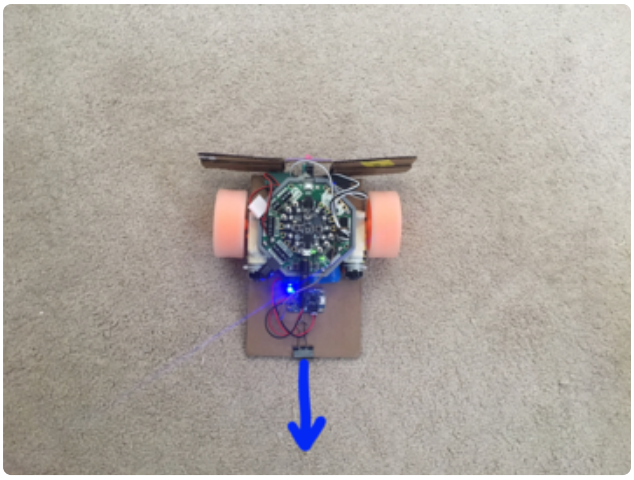


The robot will veer to the right while moving forward.

REVERSE

Hold your hand, wrist bent, with your palm facing toward you.

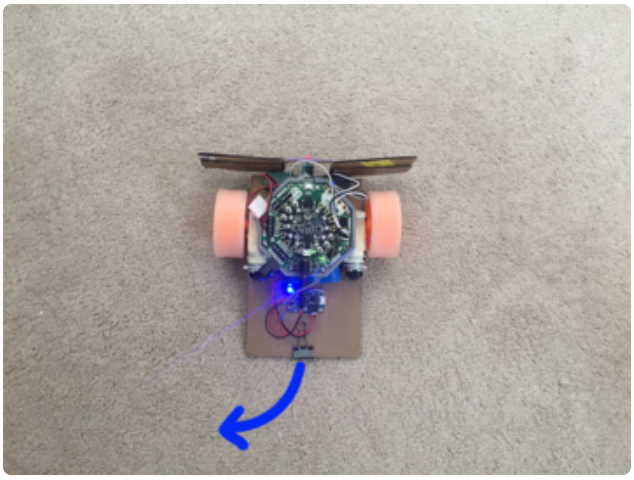
The robot will move in reverse.



Depending on the reflectivity of the surfaces in the area, the reverse commands might not be reliable.

REVERSE_LEFT

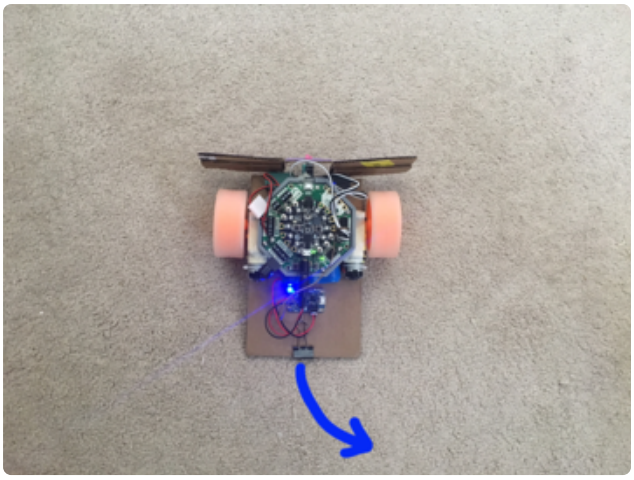
With your hand held wrist bent and palm facing you, rotate your wrist clockwise.



The robot will veer to the left while moving in reverse.

REVERSE_RIGHT

With your hand held wrist bent and palm facing you, rotate your wrist counterclockwise.



The robot will veer to the right while moving in reverse.

Controller Code

The controller code continually reads the 3 axes of the accelerometer and, based on those readings, sends the appropriate code via the IR transmitter.

Since accelerometers can detect gravity, they're great for calculating the tilt of your hand. But sometimes it hard to visualize what the motion translates to X-Y-Z values. For that, we recommend using the CircuitPython plotter

Check our our motion-plotting guide and video to see how you can read the x, y, and z accelerations and get values for use in detecting tilt.

Sensor Plotting Motion with Mu and CircuitPython

https://adafru.it/C2w

There are three primary hand positions that correspond to stop, forward, and reverse:

1. palm down: z is a high negative value,
2. hand up with palm facing away from you: y has a high positive value, and

3. hand down with palm facing toward you: y has a high negative value.

For each of these three hand positions, wrist rotation adds a turn to the basic stop/forward/reverse motion: rotating in place if the robot is stopped, or veering left/right if the robot is moving forward or reversing. The X component of the accelerometer reading indicates wrist rotation: a large negative value for a counterclockwise rotation, and a large positive value for clockwise rotation.

After configuring the hardware and libraries, the main loop reads the accelerometer and a 2 level conditional decides what should be sent to the robot.

Below is the code for the control glove. Save it to the Circuit Playground Express on the glove as `code.py`.

```
# SPDX-FileCopyrightText: 2018 Dave Astels for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import busio
import board
import pulseio
import adafruit_irremote
import adafruit_lis3dh

# Control debug output: it takes time so don't unless you're debugging
DEBUG_LOG = False

# Control codes
STOP = 0x01
ROTATE_LEFT = 0x02
ROTATE_RIGHT = 0x03
FORWARD = 0x04
FORWARD_LEFT = 0x05
FORWARD_RIGHT = 0x06
REVERSE = 0x07
REVERSE_LEFT = 0x08
REVERSE_RIGHT = 0x09

TRANSMIT_DELAY = 0.1

# Setup accelerometer
i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
sensor = adafruit_lis3dh.LIS3DH_I2C(i2c, address=0x19)

# Create a 'pulseio' output, to send infrared signals on the IR transmitter @ 38KHz
pulseout = pulseio.PulseOut(board.IR_TX, frequency=38000, duty_cycle=2 ** 15)

# Create an encoder that will take numbers and turn them into IR pulses
encoder = adafruit_irremote.GenericTransmit(header=[9500, 4500],
                                            one=[550, 550],
                                            zero=[550, 1700],
                                            trail=0)

def log(s):
    """Optionally output some text.
    :param string s: test to output
    """
    if DEBUG_LOG:
        print(s)
```

```
while True:
    x, y, z = sensor.acceleration
    log("{0: 0.3f} {1: 0.3f} {2: 0.3f}".format(x, y, z))
    if z < -5.0 and abs(y) < 3.0:         # palm down
        if x < -5.0:                      # tipped counterclockwise
            log("ROTATE_LEFT")
            encoder.transmit(pulseout, [ROTATE_LEFT] * 4)
        elif x > 5.0:                     # tipped clockwise
            log("ROTATE_RIGHT")
            encoder.transmit(pulseout, [ROTATE_RIGHT] * 4)
        else:                             # level
            log("STOP")
            encoder.transmit(pulseout, [STOP] * 4)
    elif y > 5.0:                         # palm facing away
        if x < -5.0:                      # tipped counterclockwise
            log("FORWARD_LEFT")
            encoder.transmit(pulseout, [FORWARD_LEFT] * 4)
        elif x > 5.0:                     # tipped clockwise
            log("FORWARD_RIGHT")
            encoder.transmit(pulseout, [FORWARD_RIGHT] * 4)
        else:                             # straight up
            log("FORWARD")
            encoder.transmit(pulseout, [FORWARD] * 4)
    elif y < -5.0:                        # palm facing toward (hand down)
        if x < -5.0:                      # tipped counterclockwise
            log("REVERSE_RIGHT")
            encoder.transmit(pulseout, [REVERSE_RIGHT] * 4)
        elif x > 5.0:                     # tipped clockwise
            log("REVERSE_LEFT")
            encoder.transmit(pulseout, [REVERSE_LEFT] * 4)
        else:                             #straight down
            log("REVERSE")
            encoder.transmit(pulseout, [REVERSE] * 4)

    time.sleep(TRANSMIT_DELAY)
```

Robot Code

The robot code is pretty simple: it continually reads data from the IR receiver and based on what it reads, controls the motor direction and speed.

The robot code is below. Save it to the Circuit Playground Express on the robot as `code.py`.

```
# SPDX-FileCopyrightText: 2018 Dave Astels for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import board
import pulseio
import adafruit_irremote
from adafruit_crickit import crickit

# Control debug output: it takes time so don't unless you're debugging
DEBUG_LOG = False

# Control codes
STOP = 0x01
ROTATE_LEFT = 0x02
ROTATE_RIGHT = 0x03
FORWARD = 0x04
```

```
FORWARD_LEFT = 0x05
FORWARD_RIGHT = 0x06
REVERSE = 0x07
REVERSE_LEFT = 0x08
REVERSE_RIGHT = 0x09

left_wheel = crickit.dc_motor_1
right_wheel = crickit.dc_motor_2


def log(s):
    """Optionally output some text.
    :param string s: test to output
    """
    if DEBUG_LOG:
        print(s)


# These allow easy correction for motor speed variation.
# Factors are determined by observation and fiddling.
# Start with both having a factor of 1.0 (i.e. none) and
# adjust until the bot goes more or less straight
def set_right(speed):
    right_wheel.throttle = speed * 0.9

def set_left(speed):
    left_wheel.throttle = speed


# Uncomment this to find the above factors
# set_right(1.0)
# set_left(1.0)
# while True:
#     pass


# Create a 'pulseio' input, to listen to infrared signals on the IR receiver
pulsein = pulseio.PulseIn(board.IR_RX, maxlen=120, idle_state=True)

# Create a decoder that will take pulses and turn them into numbers
decoder = adafruit_irremote.GenericDecode()

while True:
    # Listen for incoming IR pulses
    pulses = decoder.read_pulses(pulsein)

    # Try and decode them
    try:
        # Attempt to convert received pulses into numbers
        received_code = decoder.decode_bits(pulses)
    except adafruit_irremote.IRNECRepeatException:
        # We got an unusual short code, probably a 'repeat' signal
        log("NEC repeat!")
        continue
    except adafruit_irremote.IRDecodeException as e:
        # Something got distorted or maybe its not an NEC-type remote?
        log("Failed to decode: {}".format(e.args))
        continue


    if received_code == [STOP] * 4:
        log("STOP")
        set_left(0.0)
        set_right(0.0)
    elif received_code == [ROTATE_LEFT] * 4:
        log("ROTATE_LEFT")
        set_left(-0.25)
        set_right(0.25)
    elif received_code == [ROTATE_RIGHT] * 4:
```

```
        log("ROTATE_RIGHT")
        set_left(0.25)
        set_right(-0.25)
    elif received_code == [FORWARD] * 4:
        log("FORWARD")
        set_left(0.5)
        set_right(0.5)
    elif received_code == [FORWARD_LEFT] * 4:
        log("FORWARD_LEFT")
        set_left(0.0)
        set_right(0.5)
    elif received_code == [FORWARD_RIGHT] * 4:
        log("FORWARD_RIGHT")
        set_left(0.5)
        set_right(0.0)
    elif received_code == [REVERSE] * 4:
        log("REVERSE")
        set_left(-0.5)
        set_right(-0.5)
    elif received_code == [REVERSE_LEFT] * 4:
        log("REVERSE_LEFT")
        set_left(-0.25)
        set_right(-0.5)
    elif received_code == [REVERSE_RIGHT] * 4:
        log("REVERSE_RIGHT")
        set_left(-0.5)
        set_right(-0.25)
    else:
        log("UNKNOWN")
```

Wrapup

This project is pretty simple: send commands via IR based on accelerometer readings (i.e. hand position) and manipulate a CRICKIT based on commands received via IR. Even so, this can be the basis of many different projects. This just controls two motors connected to the CRICKIT, but the same approach could be applied to anything connected to the CRICKIT or Circuit Playground Express.

It could be taken further by adding simple gestures in addition to static hand position. This could be done by examining changes in accelerometer values over time.