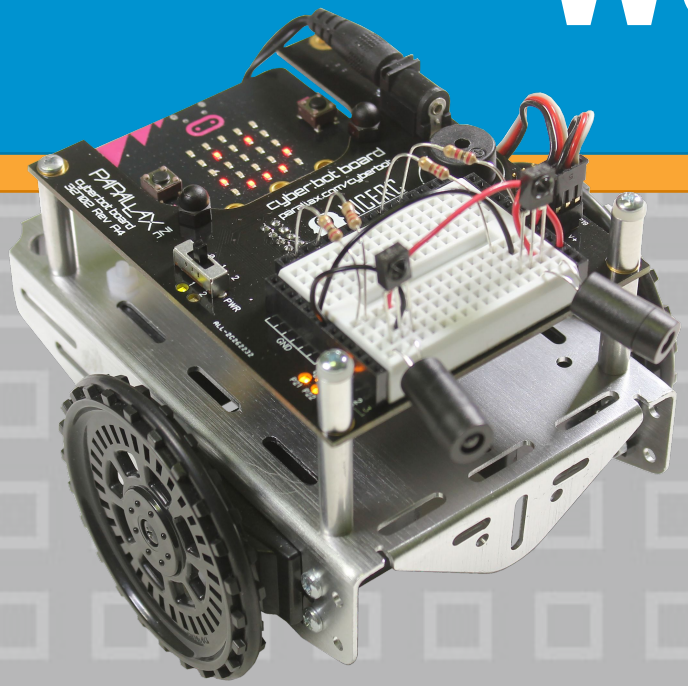
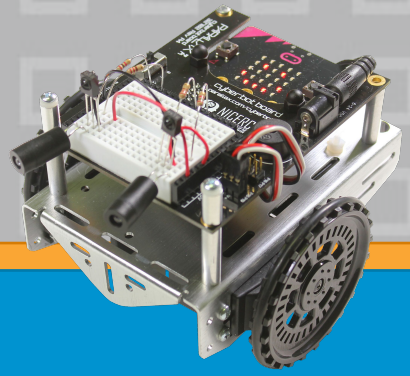
cyber:bot Educator Workshop



Supporting slides for the cyber:bot online tutorials on http://learn.parallax.com



Agenda and Goals

Agenda

- Introductions and Agenda (5 mins)
- Software (1 hour)
  - Python / MicroPython
  - micro:bit Software Setup
  - Program micro:bit
  - Add cyberbot library
- Build the cyber:bot (1 hour)
- Build circuits, coding and running our robots (5 hours)
  - LEDs and buttons
  - Touch Navigation (Whiskers)
  - Light Following Circuit
  - Infrared Object Avoidance
  - Infrared Remote Control
- Support and Products

Goals

- Enjoy the day, laugh, and play
- Use Parallax tutorials like a student
- Write our own Python code
- Break barriers about using software and hardware
- Develop basic electronic troubleshooting skills
- Understand hardware from the inside, lower levels
- Gain confidence to use cyber:bots in classrooms
- Identify other Python support materials
- See industry connections

Who really benefits?

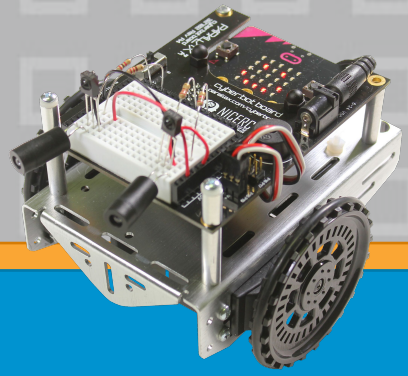
Students!

ZakUak (subscribe on YouTube) spent two days at the Parallax booth at the USA Science and Engineering Festival. He says about cyber:bot in his video:

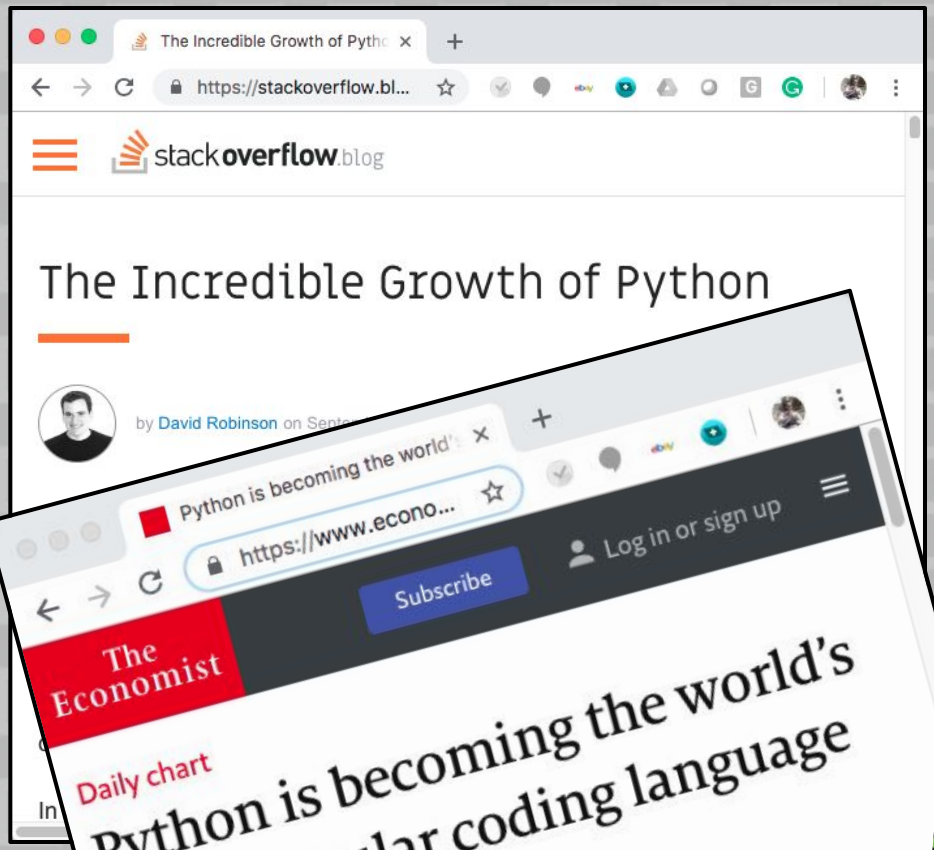
- No black box, the real thing!
- This makes coding interesting
- Python is a fun programming language

See what happens when you put these robots in class. Tell us!

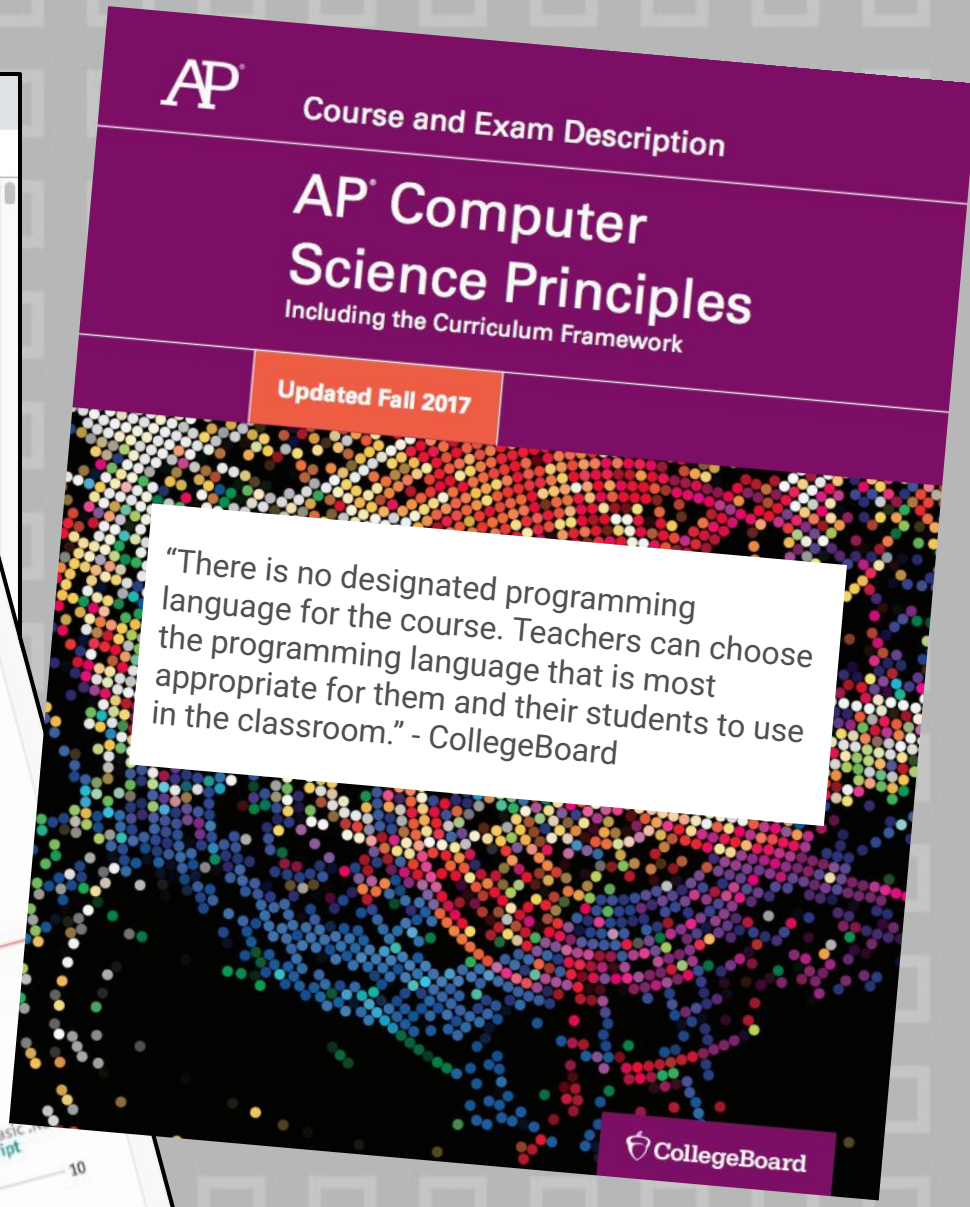
Discussion: is encouraged and appreciated!



Python and micro:bit


The Incredible Growth of Pytho
https://stackoverflow.bl...
stackoverflow.blog
The Incredible Growth of Python
by David Robinson on Sept

Python is becoming the world's
https://www.econo...
Subscribe
Log in or sign up
The Economist
Daily chart
Python is becoming the world's most popular coding language
But its rivals are unlikely to disappear
Code of conduct
Ranking of programming languages*
1988
93
98
03
08
13
18
Java (1st)
C
C++
Python
C#
Visual Basic
JavaScript
PHP
Ruby
R
Perl
Objective-C
Ada
Fortran
10
20
30


AP
Course and Exam Description
AP® Computer Science Principles
Including the Curriculum Framework
Updated Fall 2017
"There is no designated programming language for the course. Teachers can choose the programming language that is most appropriate for them and their students to use in the classroom." - CollegeBoard
CollegeBoard

https://www.zdnet.com/article/python-is-eating-the-world-how-one-developers-side-...

ZDNet

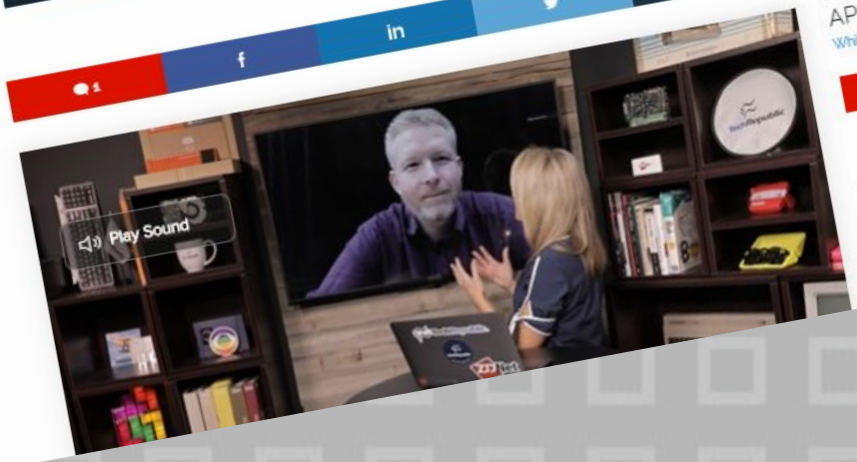
VIDEOS 5G WINDOWS 10 CLOUD AI INNOVATION SECURITY MORE NEWSLETTERS ALL WRITERS

MUST READ: Windows 10: Has Microsoft cleaned up its update mess? (Spoiler: no)

Python is eating the world: How one developer's side project became the hottest programming language on the planet

Frustrated by programming language shortcomings, Guido van Rossum created Python. With the language now used by millions, Nick Heath talks to van Rossum about Python's past and explores what's next.

By Nick Heath | July 11, 2019 -- 14:58 GMT (07:58 PDT) | Topic: Developer

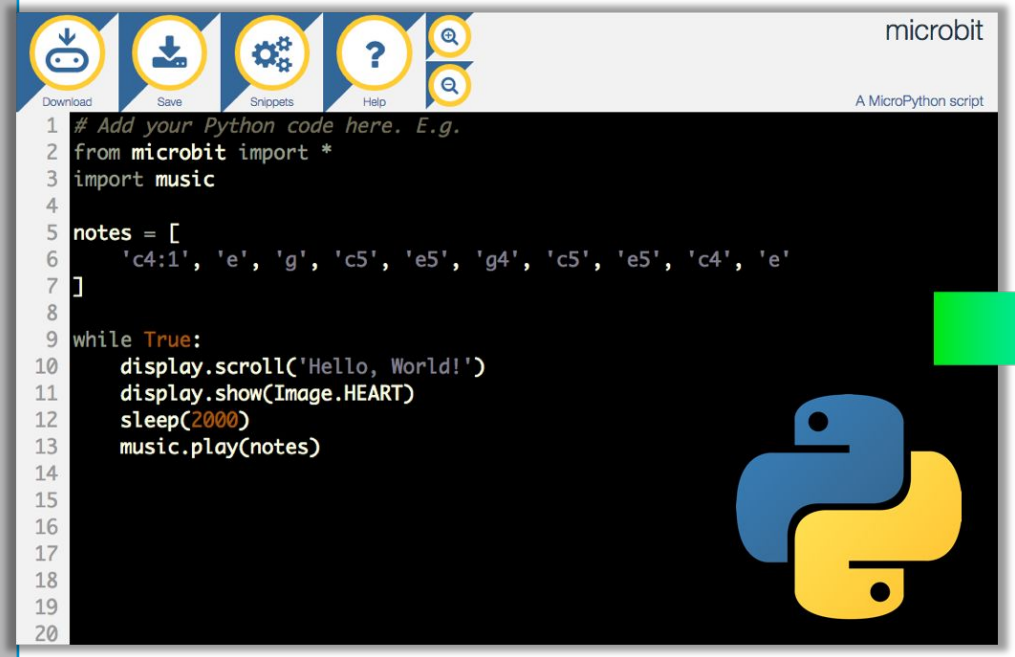


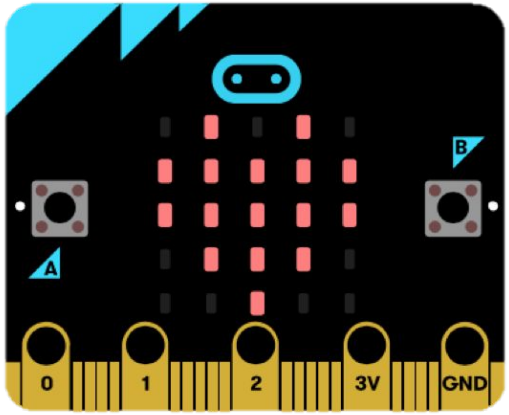
MORE FROM NICK HEATH

Windows 10
Microsoft blocks major Windows 10 update for Surface Book 2 after bug makes GPU vanish

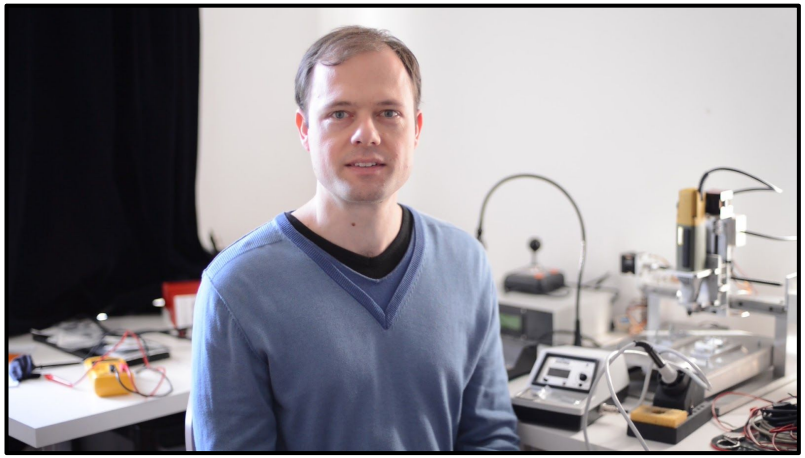
- Beginner-friendly (reads like English - looks BASIC)
- Object-oriented, structured language
- Runs on embedded hardware (MicroPython)
- Tons of examples freely available
- Programming tool support (open source, all OSs, etc.)
- Forces new programmers to use alignment/indentation for legibility (good practice)
- Not overly verbose - easier to "get at the heart" of the concept you're teaching (no wading through a bunch of meaningless syntax rules that obscure the instructional intent)
- Free / open source (no awkward licensing/copyright)

MicroPython - Python for Microcontrollers

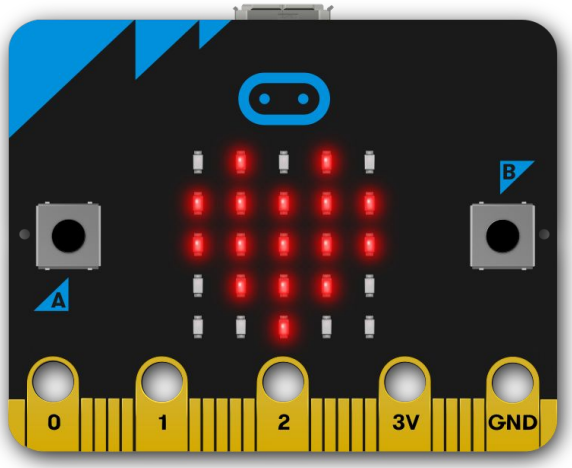




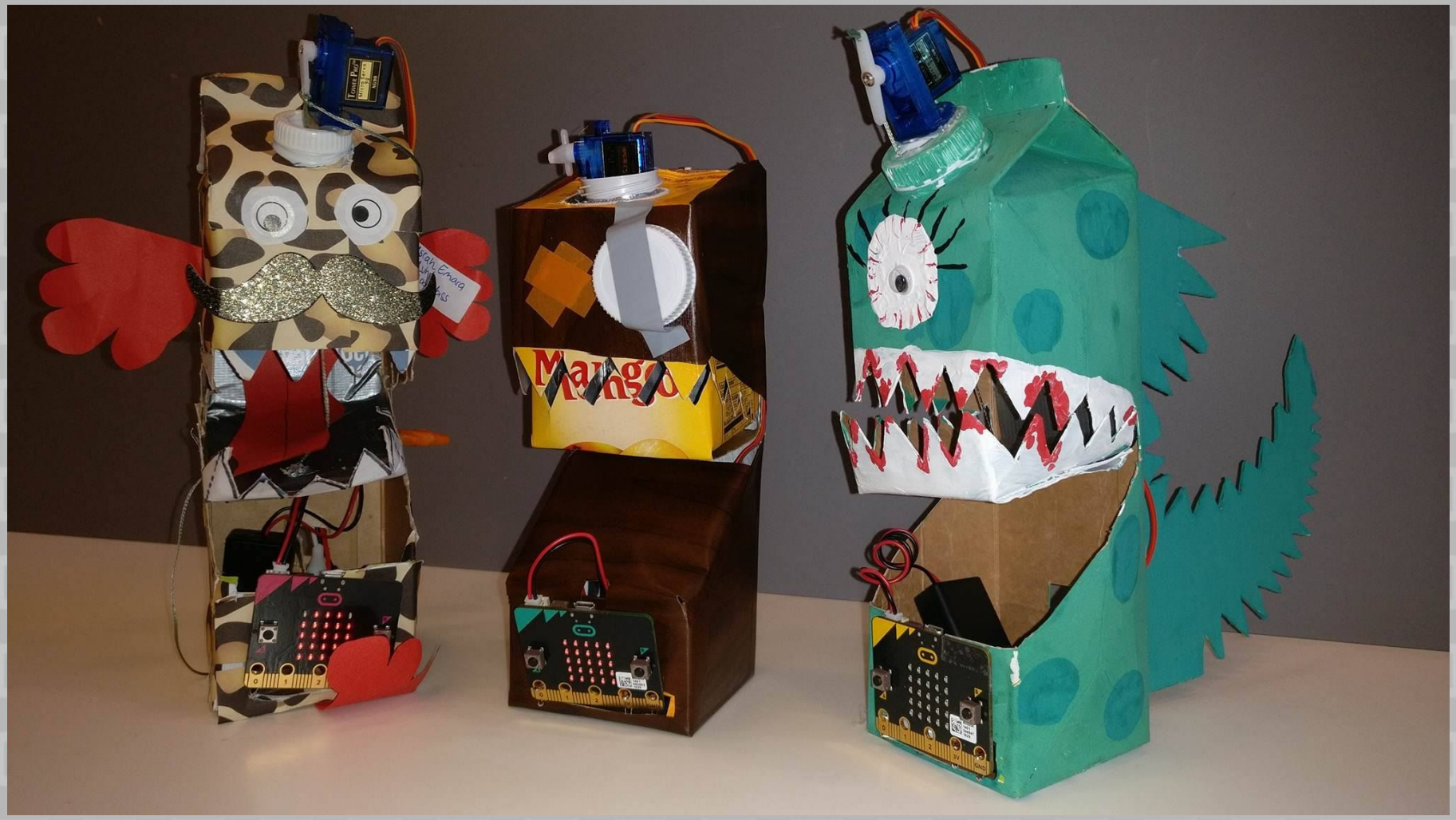
MicroPython is a creation of Damien George



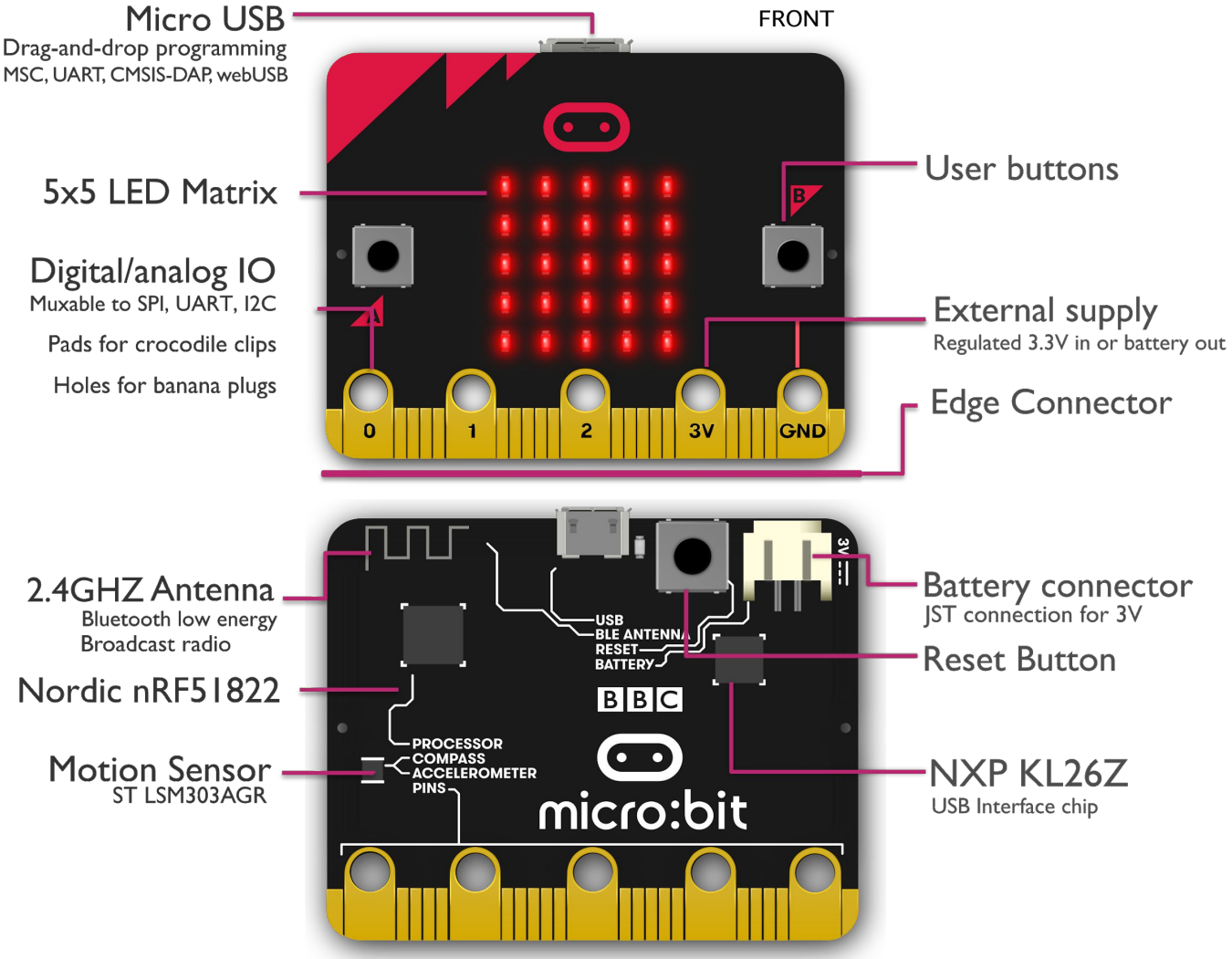
micro:bit


A
B
0
1
2
3V
GND


FCC ID: 2AKFPMB013B
USB
BLE ANTENNA
RESET
BATTERY
PROCESSOR
COMPASS
ACCELEROMETER
PINS
micro:bit
V1.3B
3V
BBC

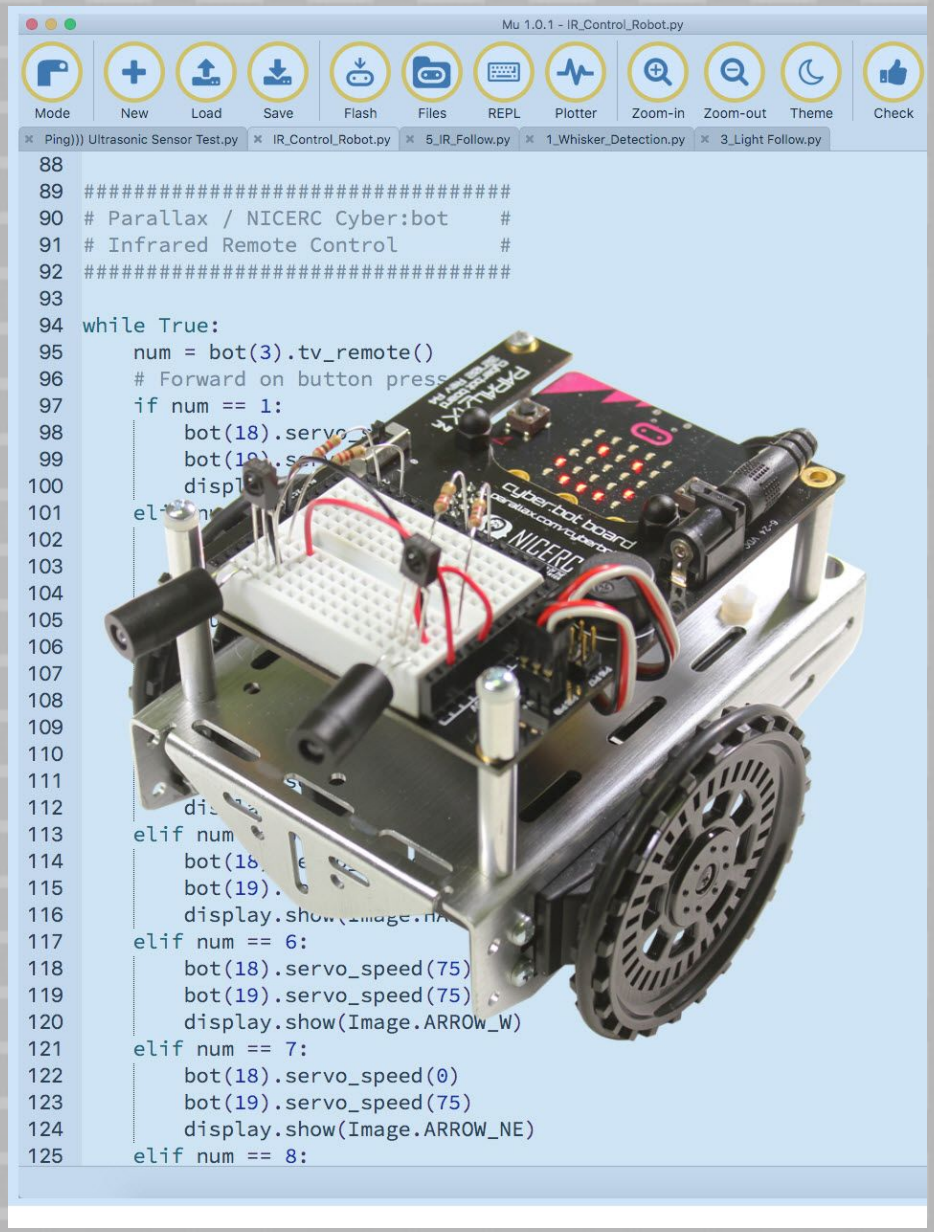

Mango

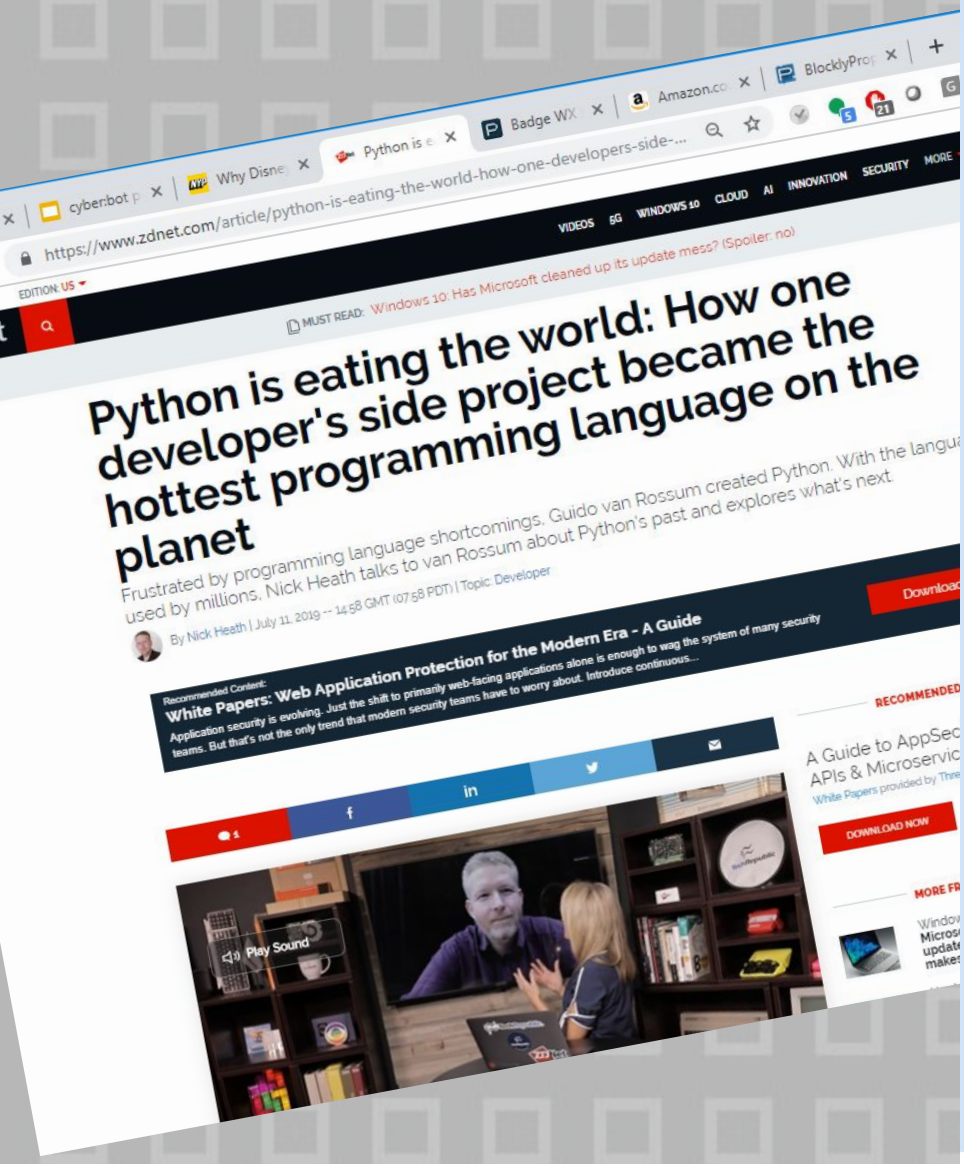
micro:bit: Hardware Features

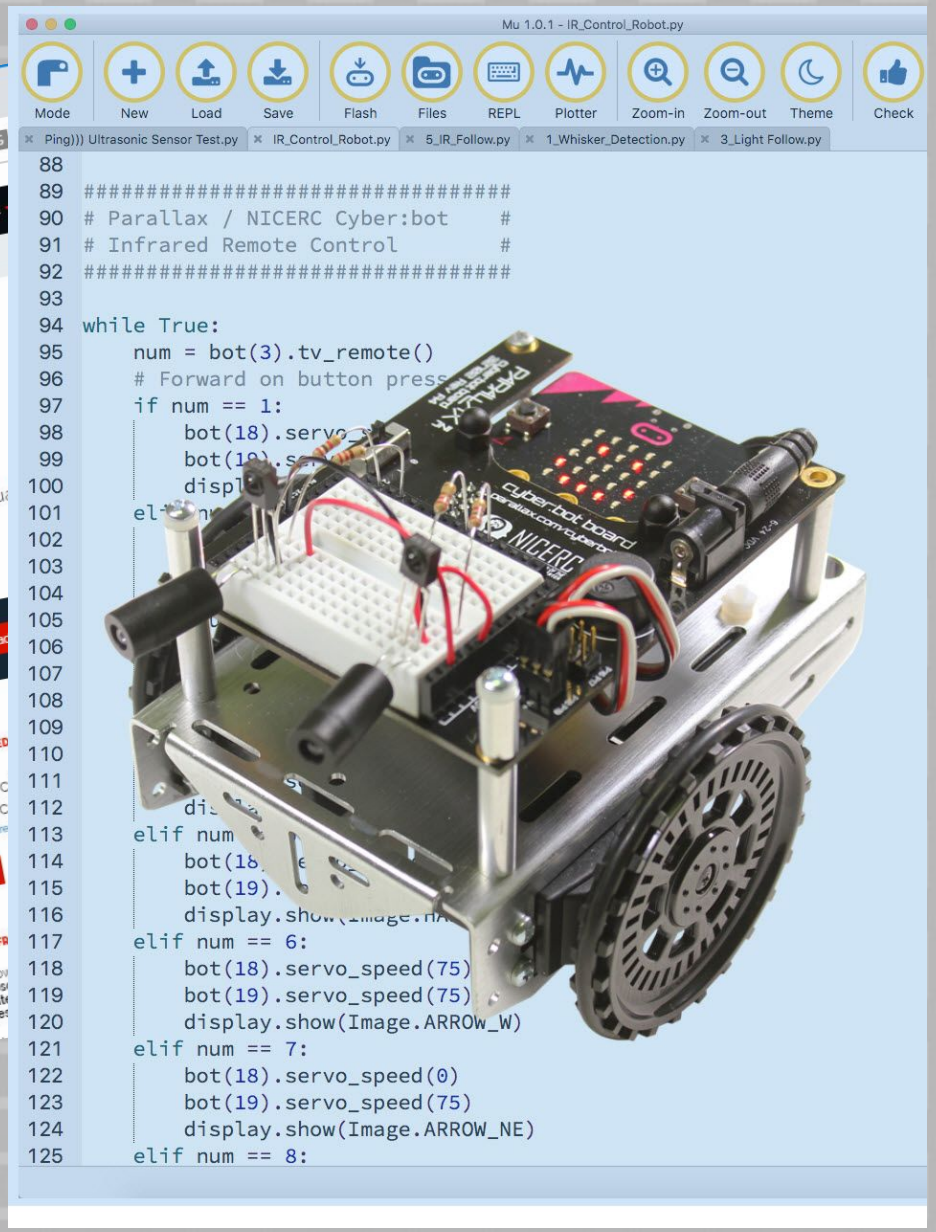


Python + Robotics

- Students learn more when they can “see” their programs run
- Competition-based challenges make it fun
- Basis for learning product development, robotics, and mechanical
- Low-level skills further creative vs. user-level experience



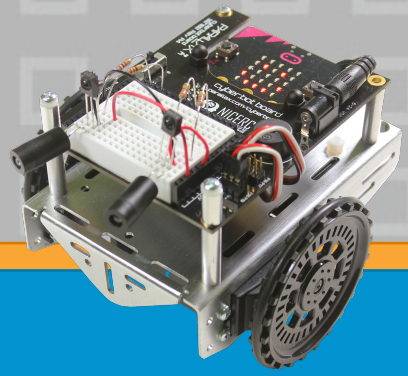

Python is eating the world: How one developer's side project became the hottest programming language on the planet
Frustrated by programming language shortcomings, Guido van Rossum created Python. With the langua
used by millions, Nick Heath talks to van Rossum about Python's past and explores what's next.
By Nick Heath | July 11, 2019 -- 14:58 GMT (07:58 PDT) | Topic: Developer
Recommended Content:
White Papers: Web Application Protection for the Modern Era - A Guide
Application security is evolving. Just the shift to primarily web-facing applications alone is enough to wag the system of many security teams. But that's not the only trend that modern security teams have to worry about. Introduce continuous...
RECOMMENDED
A Guide to AppSec APIs & Microservic
White Papers provided by Thre
DOWNLOAD NOW
Play Sound


Mu 1.0.1 - IR_Control_Robot.py
Mode
New
Load
Save
Flash
Files
REPL
Plotter
Zoom-in
Zoom-out
Theme
Check
Ping))) Ultrasonic Sensor Test.py
IR_Control_Robot.py
5_IR_Follow.py
1_Whisker_Detection.py
3_Light Follow.py
88
89 ##################################
90 # Parallax / NICERC Cyber:bot   #
91 # Infrared Remote Control   #
92 ##################################
93
94 while True:
95   num = bot(3).tv_remote()
96   # Forward on button press
97   if num == 1:
98   bot(18).servo_
99   bot(19).s
100   displ
101   el
102
103
104
105
106
107
108
109
110
111
112   di
113   elif num
114   bot(18
115   bot(19).
116   display.show
117   elif num == 6:
118   bot(18).servo_speed(75)
119   bot(19).servo_speed(75)
120   display.show(Image.ARROW_W)
121   elif num == 7:
122   bot(18).servo_speed(0)
123   bot(19).servo_speed(75)
124   display.show(Image.ARROW_NE)
125   elif num == 8:

cyber:bot is a joint project of NICERC and Parallax

- https://nicerc.org/
- Department of Homeland Security funded
- Create a cyber-ready workforce
- Produces a free curriculum for American educators
- Staff of 25

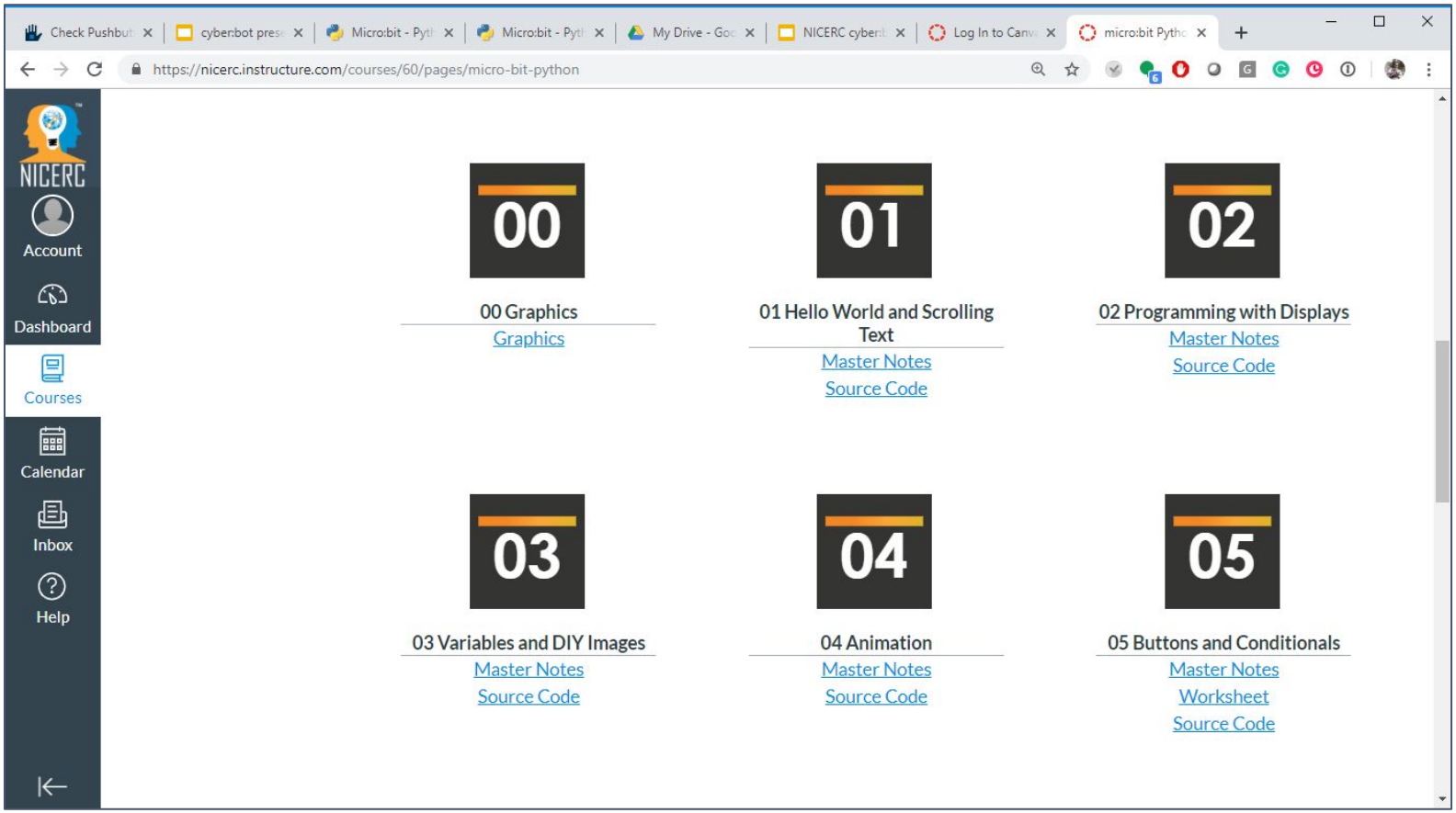
- https://www.parallax.com
- Established educational robotic company since 1992
- American manufacturer from Rocklin, California
- Staff of 25



Educational Resources

NICERC.org Cyber Fundamentals

Create a login for their micro:bit Python lessons

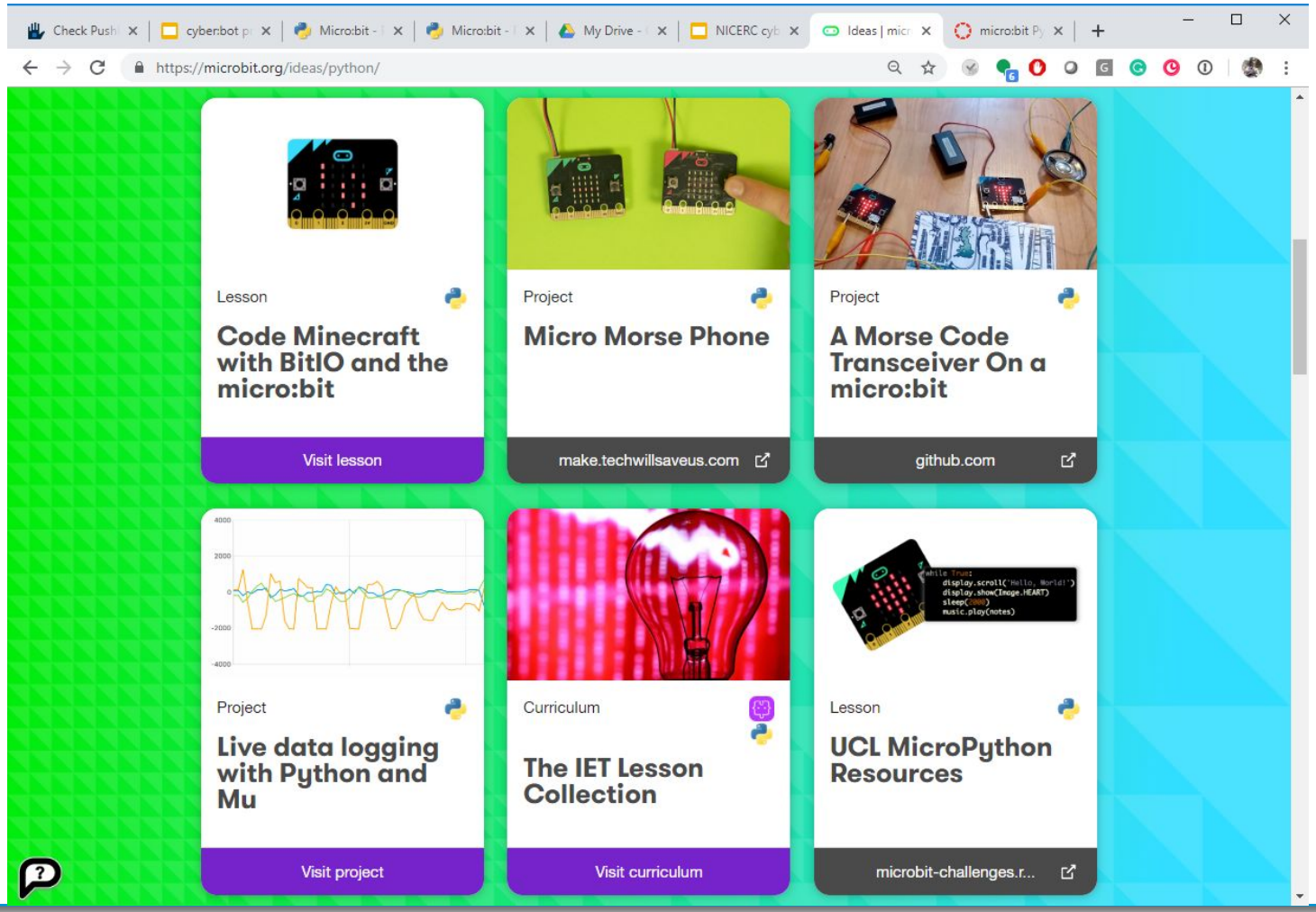


NICERC.org Cyber Fundamentals

Lesson	Title
00	Graphics
01	Hello World and Scrolling Text
02	Programming with Displays
03	Variables and DIY Images
04	Animation
05	Buttons and Conditionals
06	Compass and Comparisons
07	Binary and Visual Counter
08	Tug-o-War
09	Communications
10	Student Response System
11	Passwords and Security
12	Voltage Measurement
13	Temperature Sensor

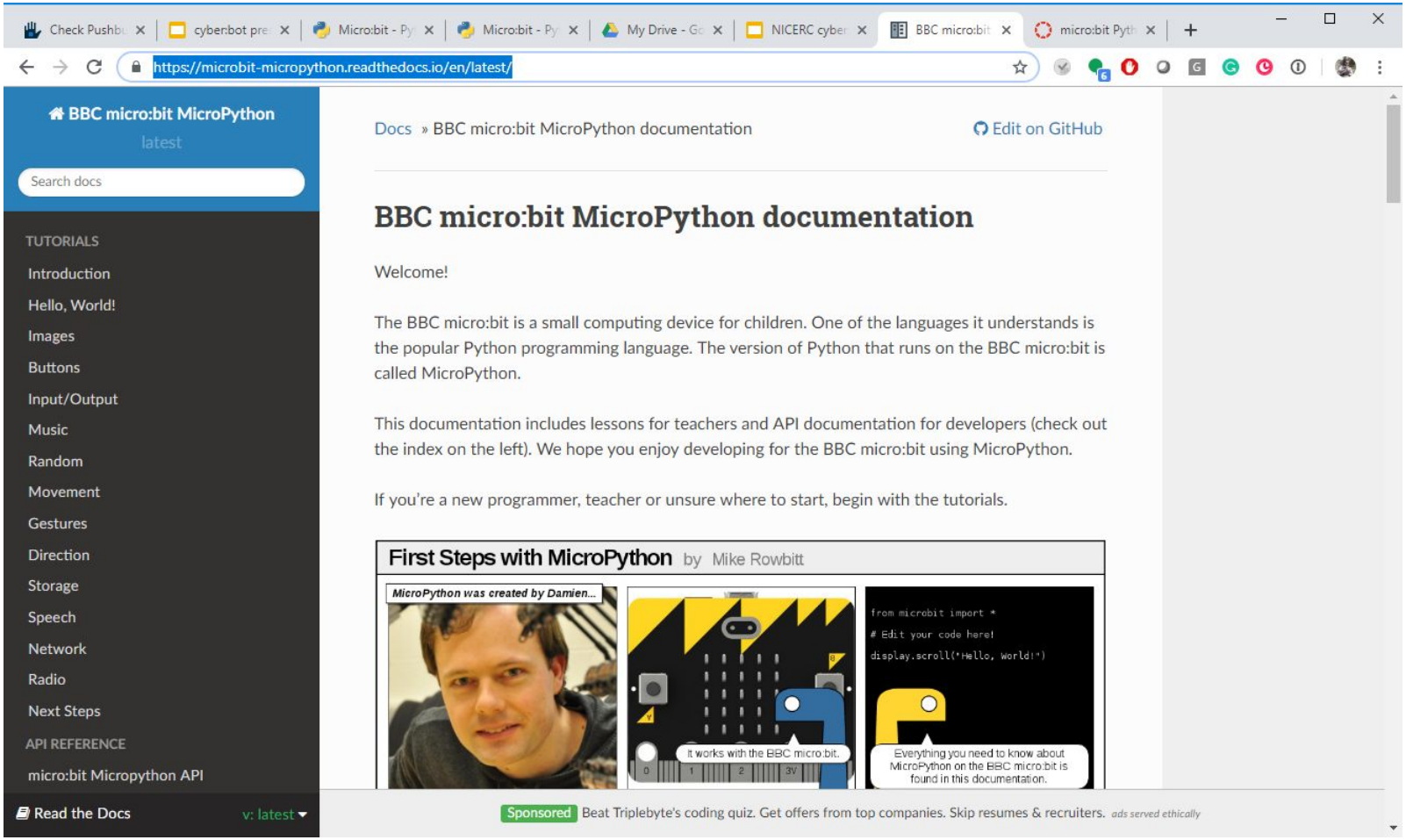
micro:bit “Ideas”

https://microbit.org/ideas/python/



BBC micro:bit Python Documentation

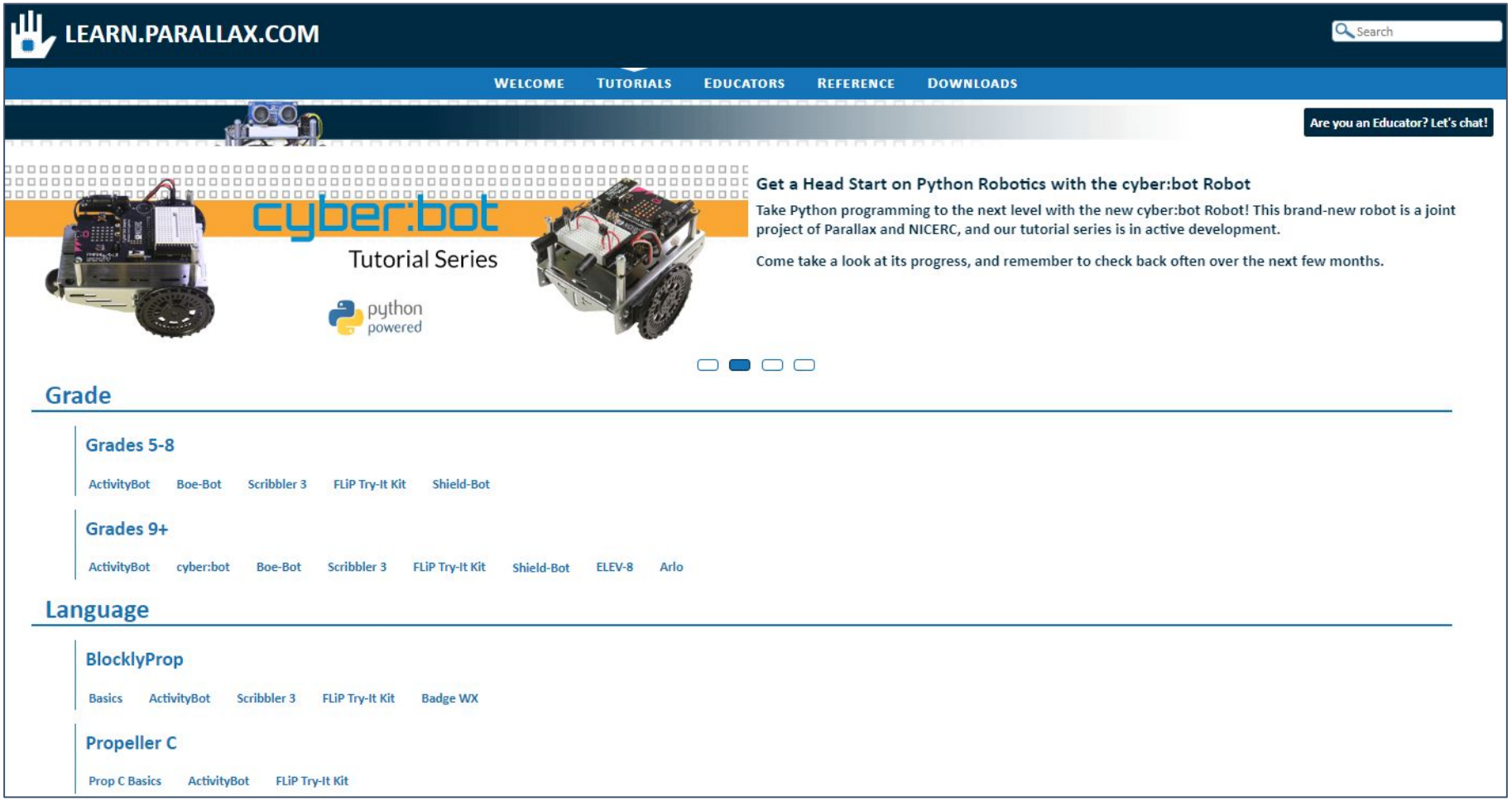
https://microbit-micropython.readthedocs.io/en/latest/

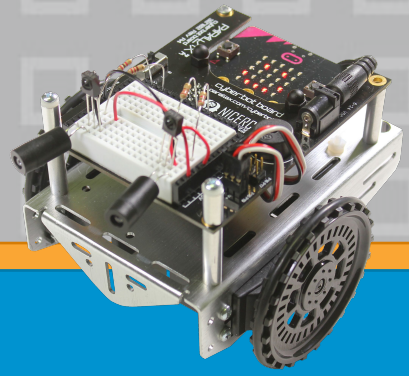


Hackster.io Educator Resources

The Hardware (1), JavaScript (2) and Python (3)

Learn.parallax.com

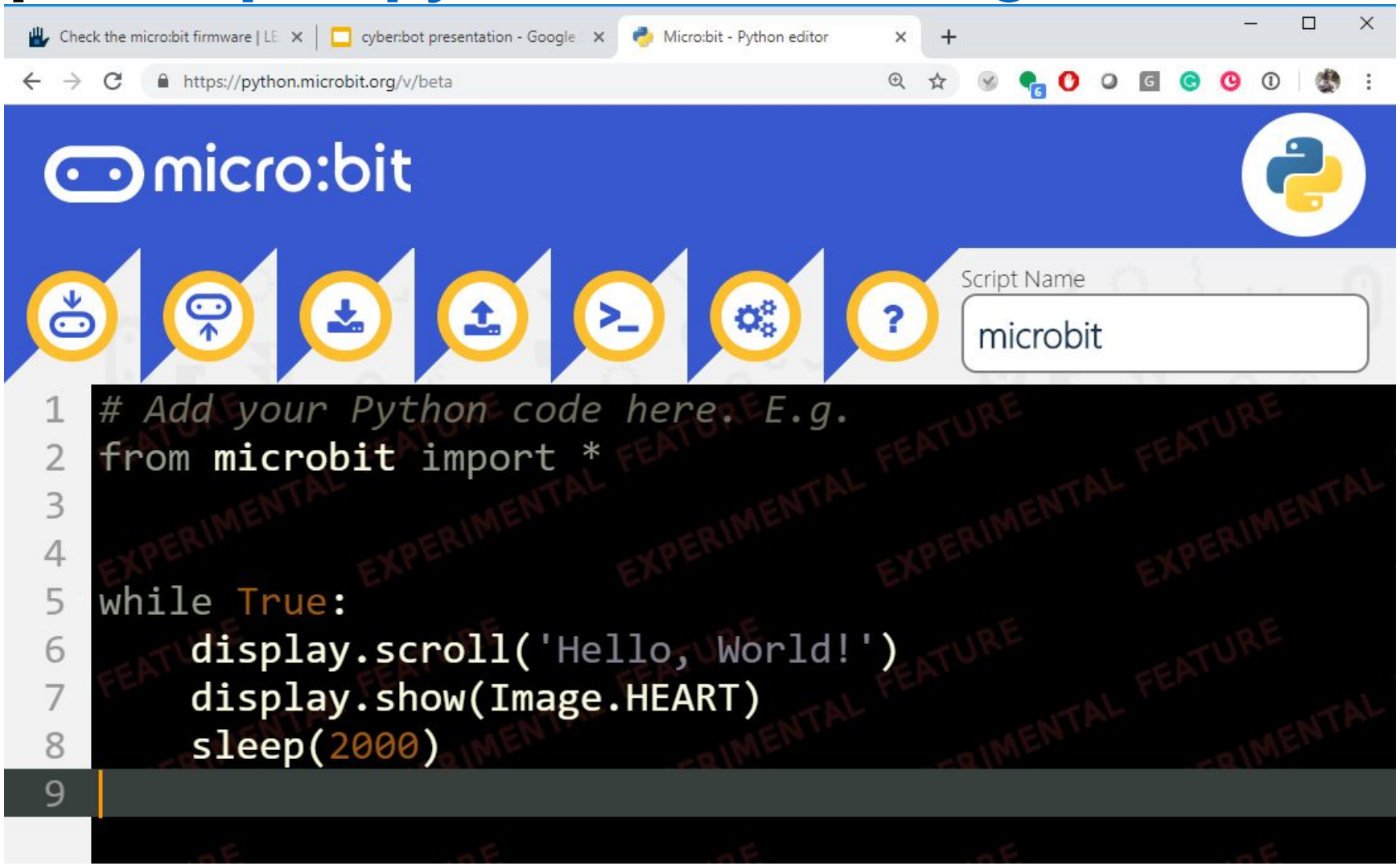


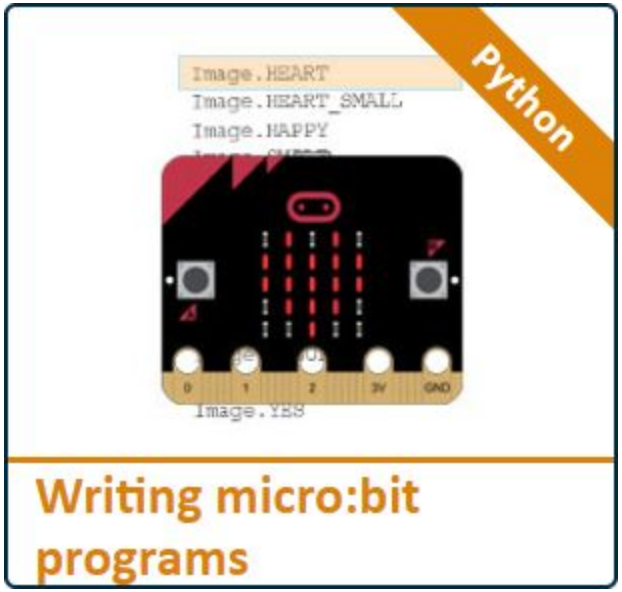


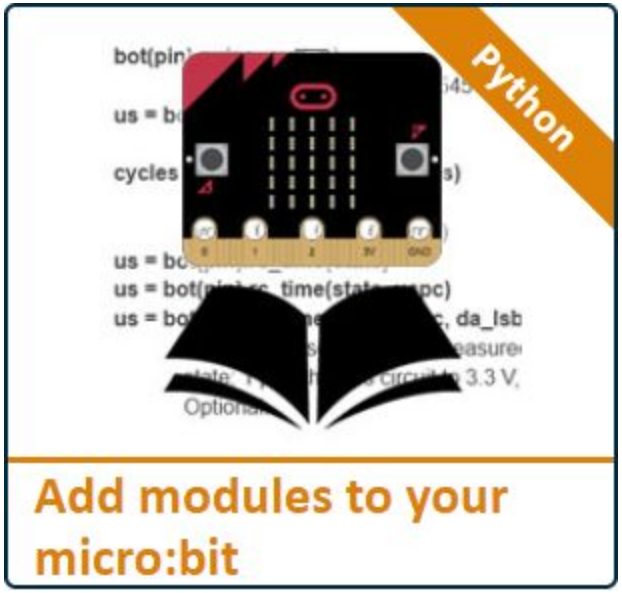
Software Setup

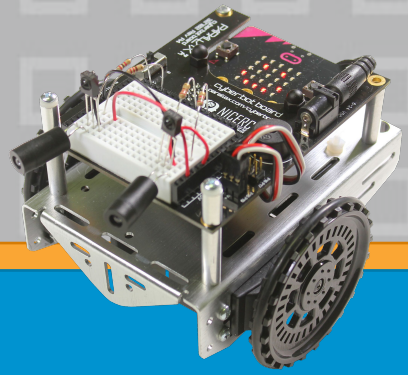
Connect micro:bit to Computer

Open https://python.microbit.org/v/beta









Writing micro:bit Programs

LED Matrix - Scrolling

```
#hello_goodbye.py

from microbit import *

display.scroll('Hello', delay = 500)
display.scroll('Goodbye', delay = 150)
```

- Change the text with your own message.
- Change the delays.
- Add more lines of code and Flash!

LED Matrix - Premade Images

```
#hello_goodbye.py

from microbit import *

display.show(Image.HEART)
sleep (500)
display.show(Image.HEART_SMALL)
sleep (500)
```

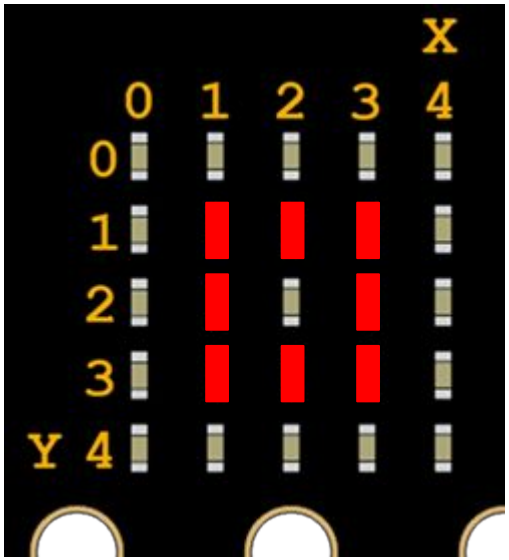
- Press the “reset” button to see it again.
- Try your own - Google “micro:bit MicroPython Images”

Custom Images

`display.set_pixel (x, y, b)`

`x` and **`y`** are pixel position, **`b`** is brightness

```
# medium_box.py

from microbit import *

display.set_pixel(1, 1, 9)
display.set_pixel(1, 2, 9)
display.set_pixel(1, 3, 9)
display.set_pixel(2, 1, 9)
display.set_pixel(2, 3, 9)
display.set_pixel(3, 1, 9)
display.set_pixel(3, 2, 9)
display.set_pixel(3, 3, 9)
```



- Draw your own shape.
- Change the brightness.

Solve Math Problems

- assignment (=)
- addition (+)
- subtraction (-)
- multiplication (*)
- division (/)

```
#simple_math.py

from microbit import *

a = 89
b = 42
c = b + a

display.scroll("a = b = ")
display.scroll(c)
```

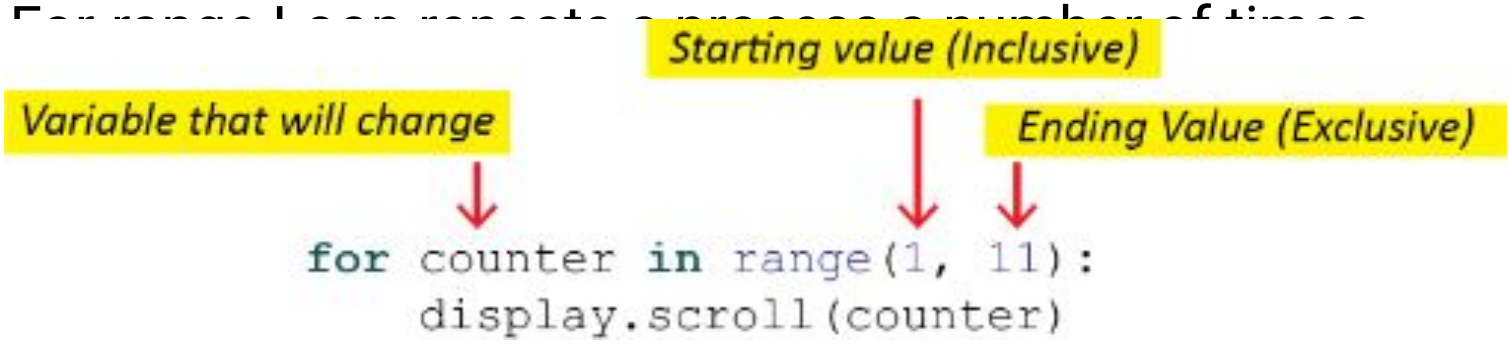
Make Decisions - If Else Statement

- compare-equals (==)
- greater than (>)
- less than (<)
- greater than or equal to (>=)
- less than or equal to (<=)

```
#simple_decision.py

from microbit import *

a = 89
b = 42
if a > b:
   display.scroll("a is greater than b")
else:
   display.scroll("a is not greater than b")
```

Count and Repeat

- Many robotic tasks involve repeating an action over and over.
- For range Loop repeats a process a number of times




```
#count_to_ten.py

from microbit import *

for counter in range (1, 11):
    display.scroll(counter)

display.scroll("All done!")
```

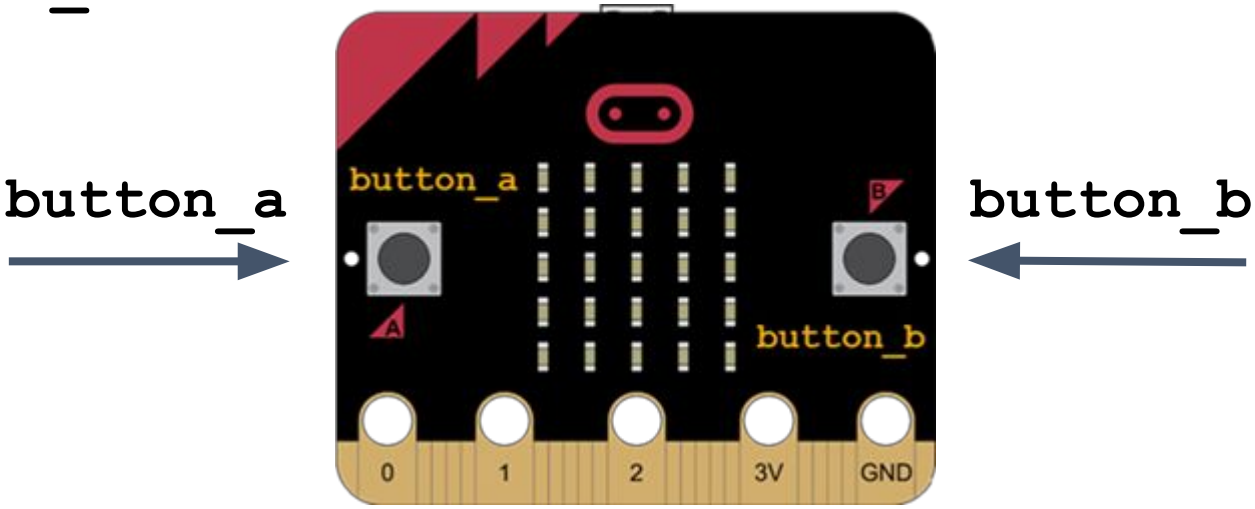
Constants and Comments

- " " " is for block comments (docstrings) and allow many lines to be commented as documentation.
- # is for line comments and explains subsequent line of code

```
"""
count_to_ten_documented.py
"""

from microbit import *              #import the microbit library

START_VAL = 1                       #set START_VAL to 1
STOP_VAL = 11                       #set STOP_VAL to 11
STEP_VAL = 1                        #set STEP_VAL to 1

#count from START_VAL to STOP_VAL incrementing by STEP_VAL
for counter in range(START_VAL, STOP_VAL, STEP_VAL):
    display.scroll(counter)         #display the counter value

display.scroll("All Done!")         #display message when done
```

Pushbuttons

- The micro:bit module can be programmed to respond to these buttons being pressed.
- Each button exists as an object referred to as **`button_a`** and **`button_b`**.

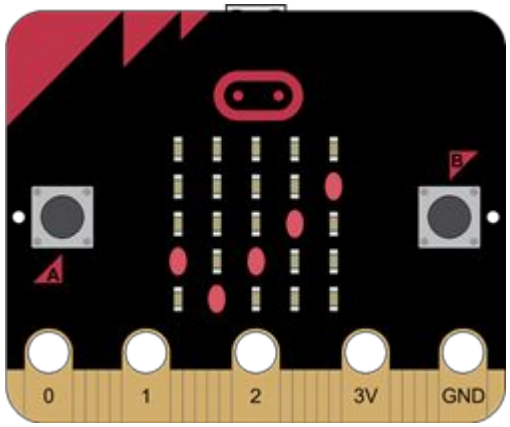


- Used to change program functions, start/stop, or play games.
- These are not the reset button, but there is a reset button too!

Pushbuttons

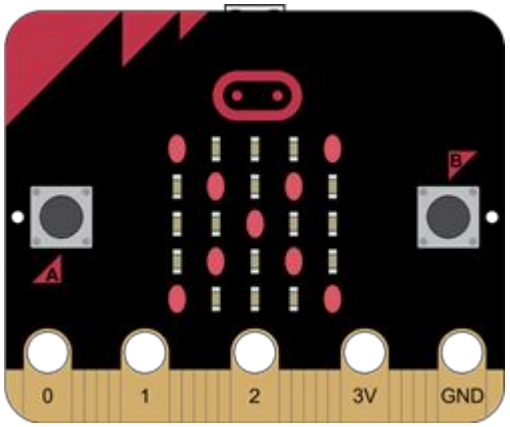
is_pressed()

```
#is_pressed.py
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.YES)
    else:
        display.show(Image.NO)
```



was_pressed()

```
#was_pressed.py
from microbit import *

while True:
    sleep(5000)
    if button_a.was_pressed():
        display.show(Image.YES)
    else:
        display.show(Image.NO)
```



Modules, Methods, Functions, and Objects

- *Modules* are code libraries that include the *objects*
- *Methods* are *functions* that belong to a specific *object*
- *Functions* are defined by a def statement. Functions can pass parameters (arguments) - like robot speed, sensor states.

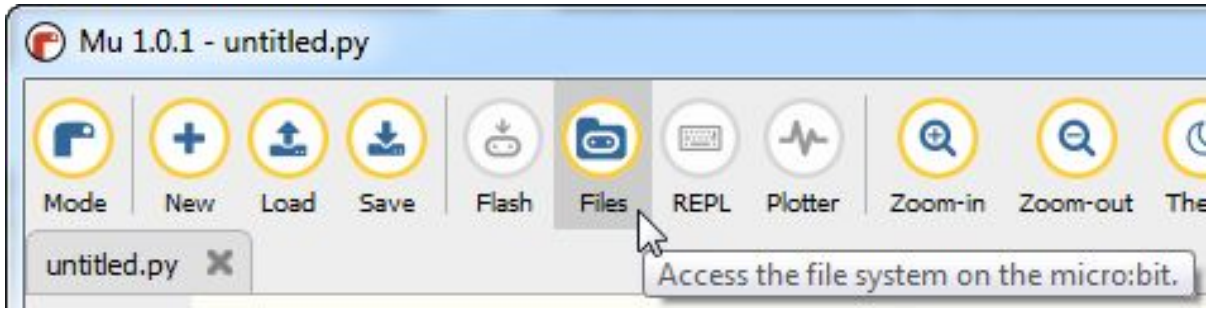

```
#smile.py

from microbit import *

def smile():
    display.show(Image.HAPPY)
    sleep(2000)

smile()
```

Adding the cyberbot library (module)

- Two editor choices: online Python editor (https://python.microbit.org/v/beta) or local installed Mu editor (https://codewith.mu/)
- Use the online Python editor (less trouble with tabs/spaces)

online Python editor

local install Mu

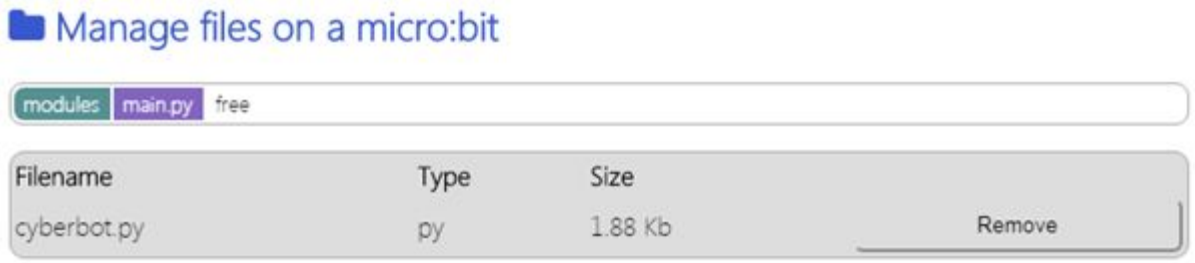


Adding the cyberbot library (module)

1. Download the cyberbot library archive to your desktop from https://bit.ly/2XOgvGk or “Add modules to your micro:bit”
2. Extract the contents of the file to a new folder.
3. “Load” Python code or hex file using Load button

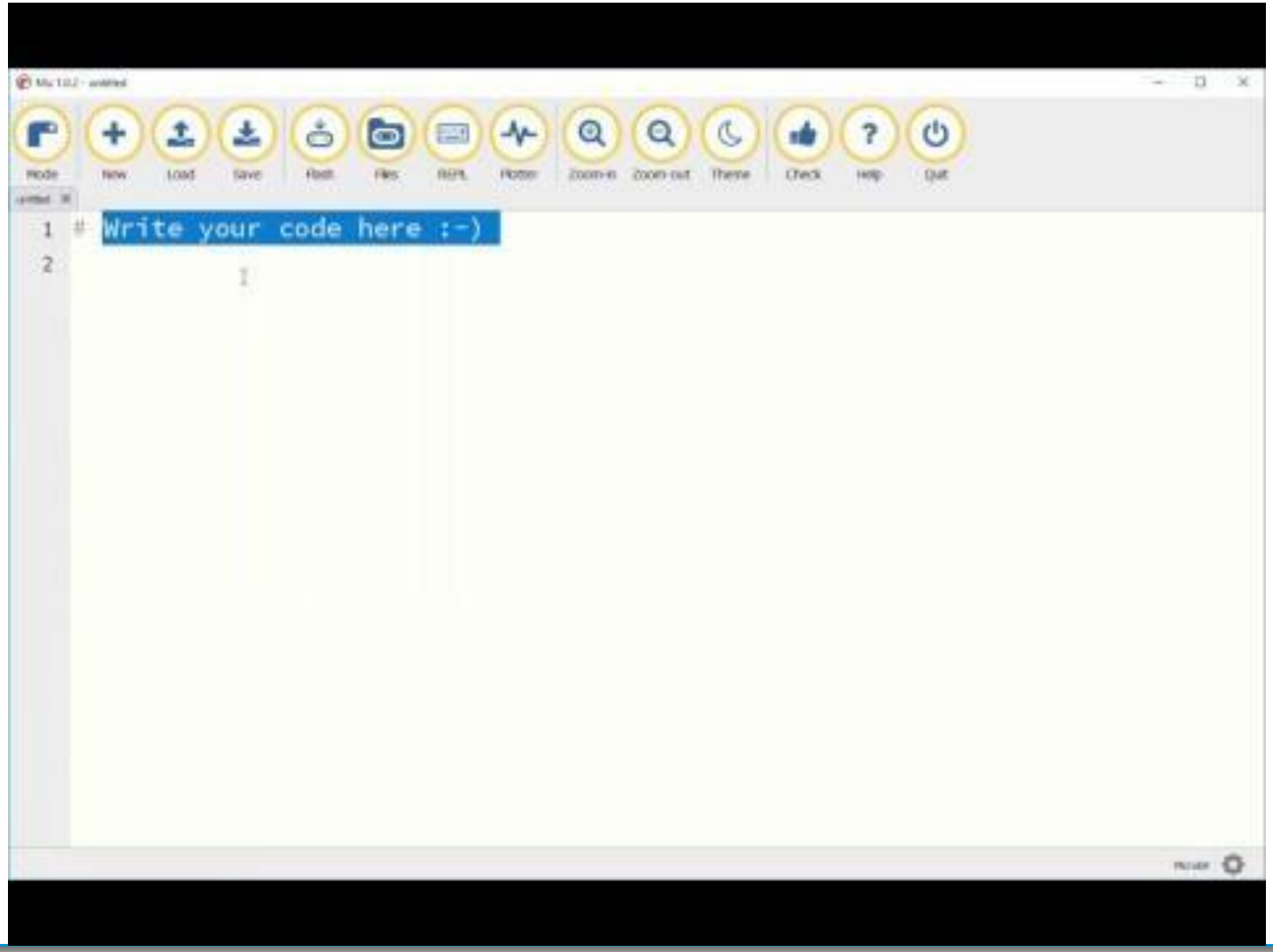
Adding the cyberbot library (module)

4. Check that the **cyberbot.py** file appears.



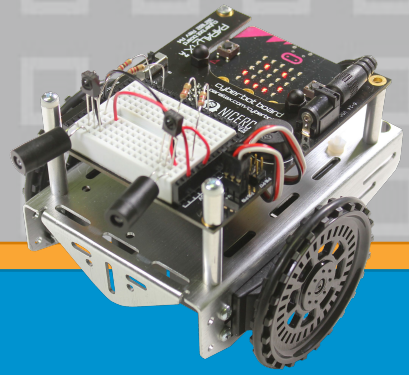
5. Close the window.

Adding cyberbot library with Mu editor



Cyberbot.py library contents

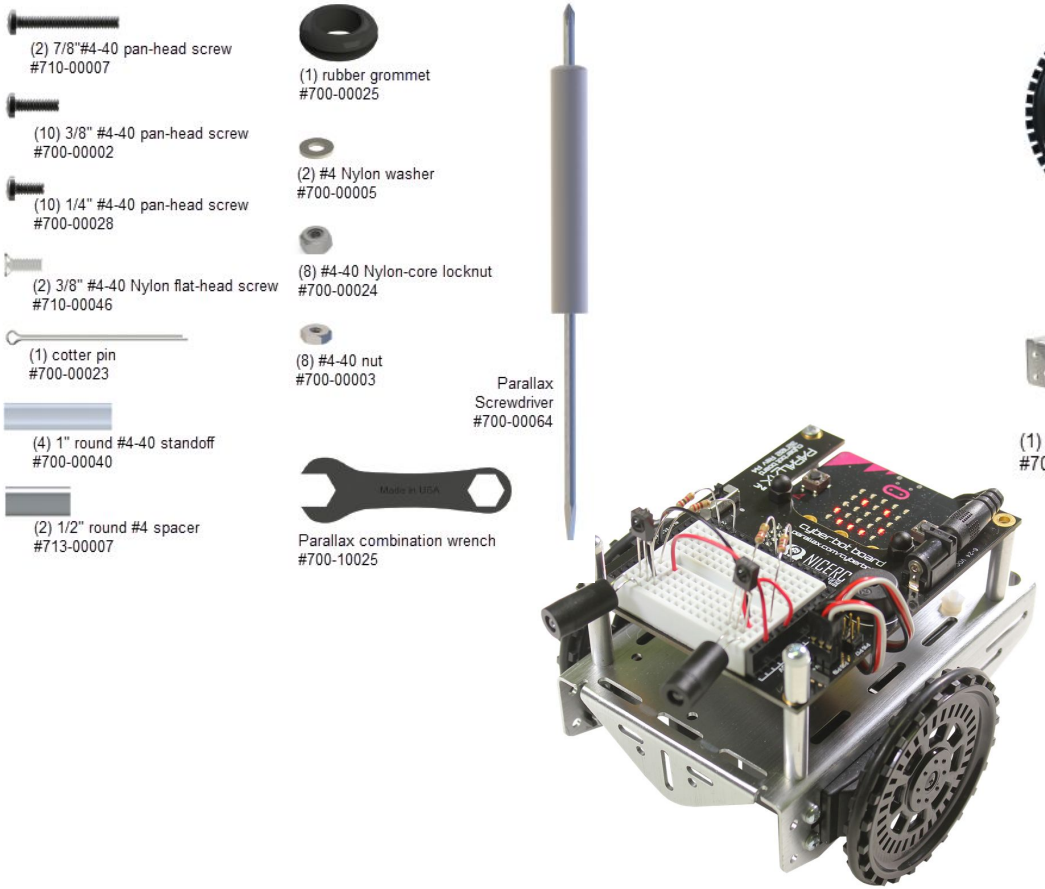
```
bot(pin).read_r(data)                # retrieve returned value via I2C
bot(pin).digital_write(state)        # set I/O pins high or low
bot(pin).analog_write(PWM)           # set duty cycle to four available PWM channels
bot(pin).digital_read(state)         # get I/O pin state high or low
bot(pin).states(states)              # set binary pin states to multiple I/Os
bot(pin).directions(directions)      # set I/O pin directions
bot(pin).qti(QTI values)             # set and read four line follower sensors
bot(pin).pulse_out(pulsewidth)       # set and maintain a pulse
bot(pin).pulse_in(pulsewidth)        # measure pulse on I/O pin (accelerometers)
bot(pin).pulse_count(counts)         # count pulses over duration of time
bot(pin).rc_time(time)               # pseudo-analog R/C charge/discharge time on I/O pin
bot(pin).frequency_out(sound)        # set frequency, duration to I/O pin
bot(pin).ir_detect(frequency)        # generate IR pulse and get receiver value
bot(pin).servo_angle(angle)          # set and hold servo in an angle (up to 14 servos)
bot(pin).servo_speed(speed)          # set and hold servo speed (-100 to 100)
bot(pin).servo_disable(disable)      # disable a servo
bot(pin).ping_distance(distance)     # configure Ultrasonic or Laser Ping, receive distance
bot(pin).tv_remote(button)           # decode pulses from Sony TV remote and return button number
```

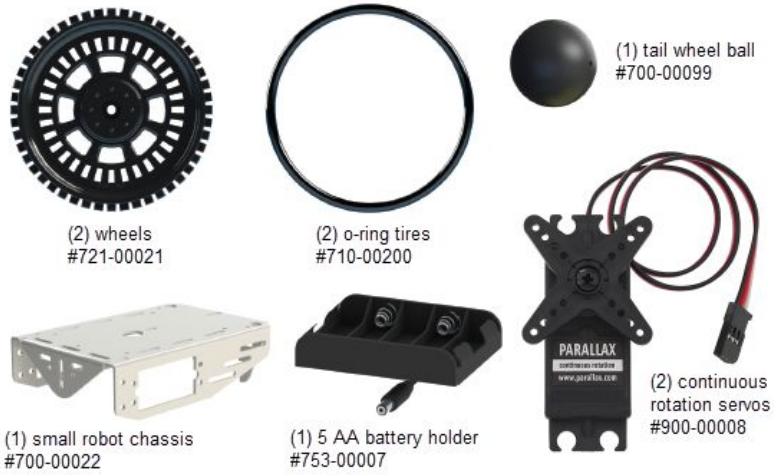


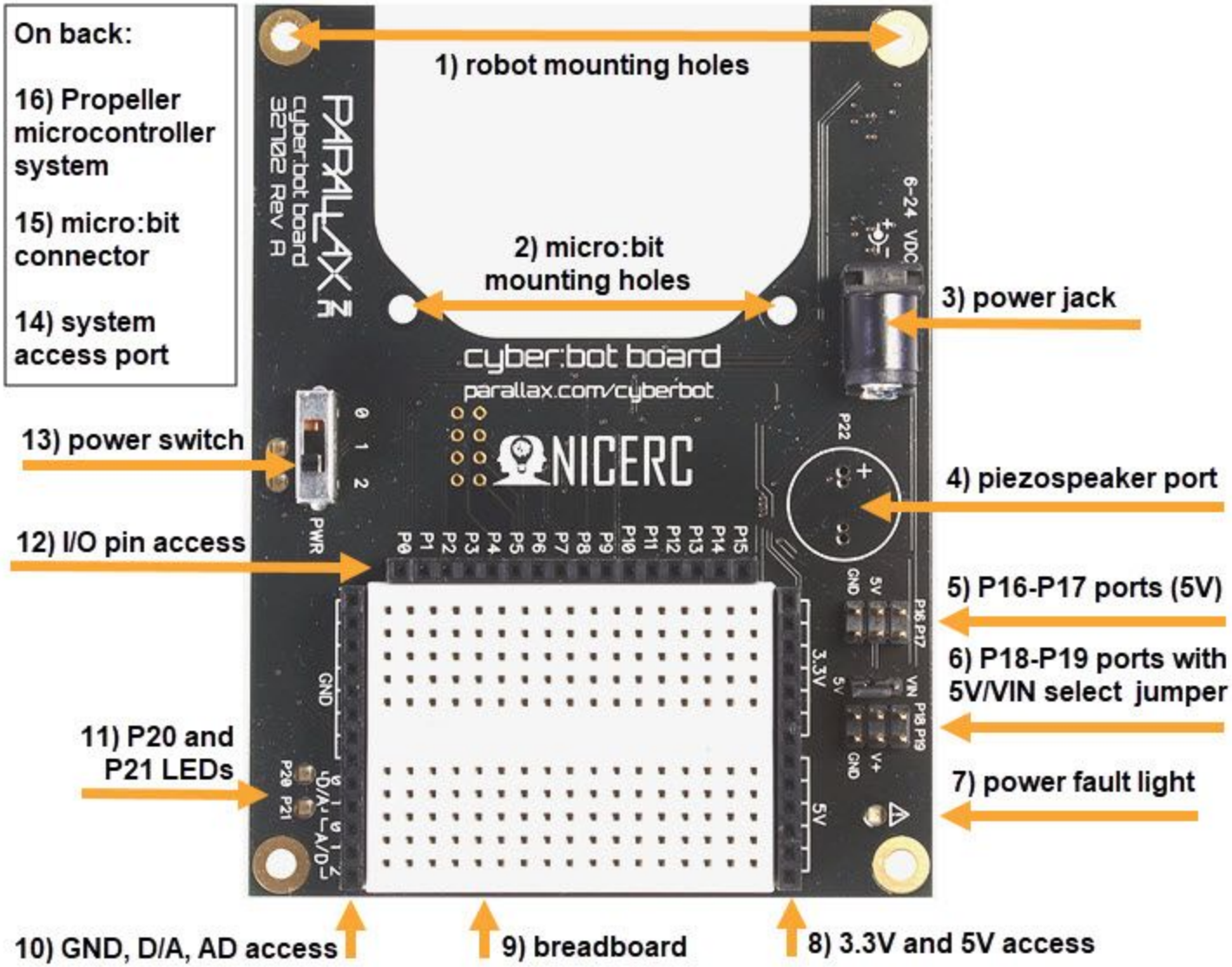
Build your cyber:bot

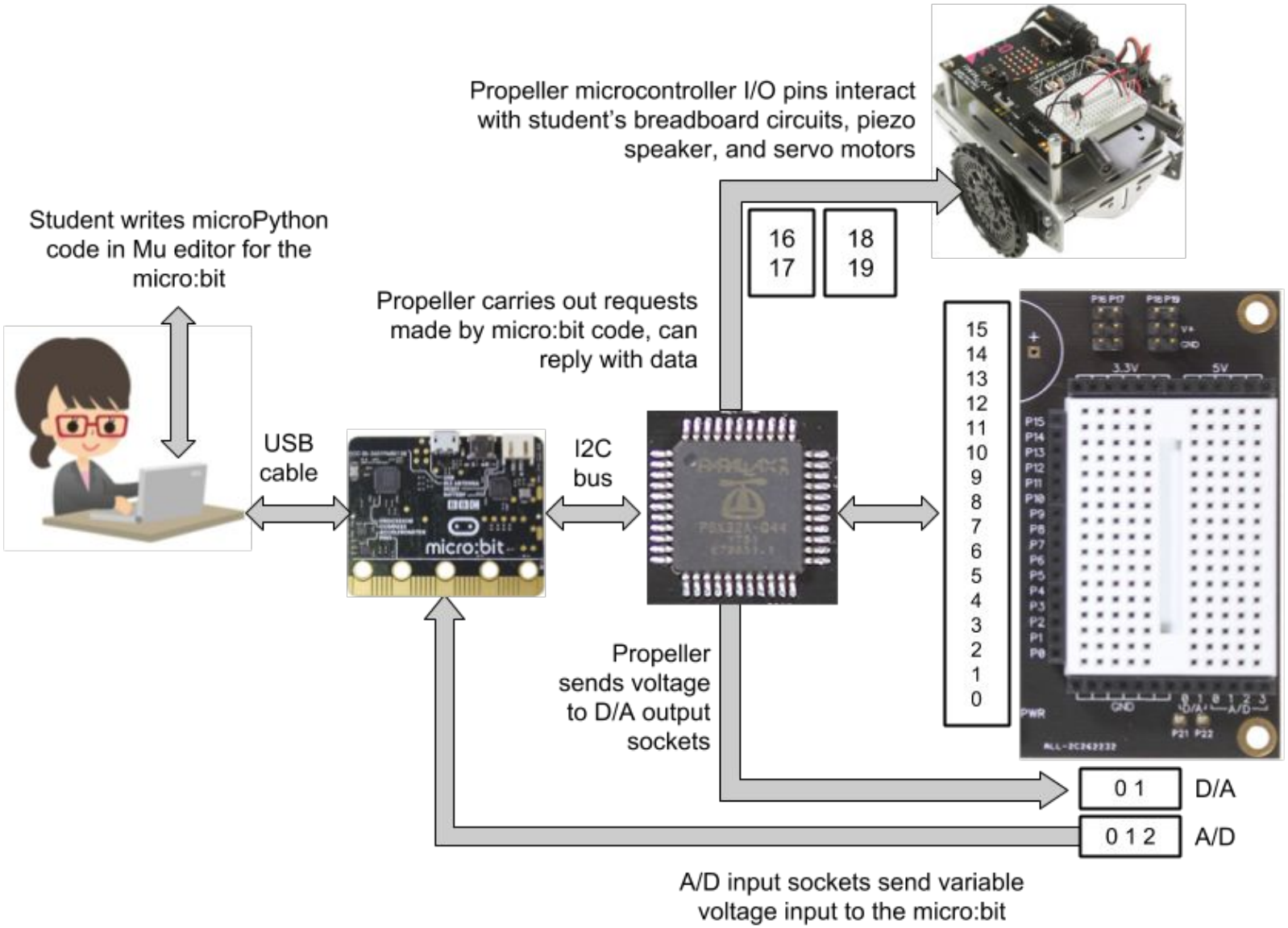
Assemble your cyber:bot

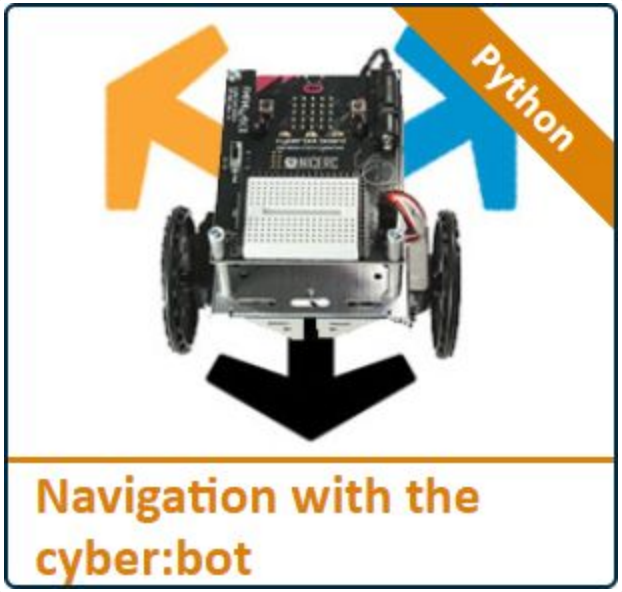
https://learn.parallax.com

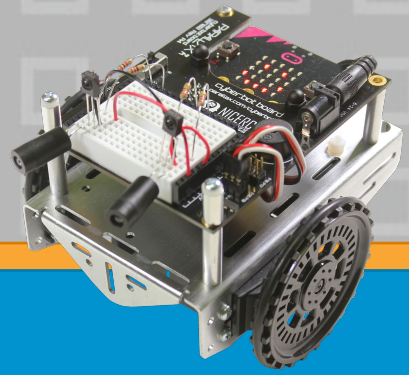





On back:
16) Propeller microcontroller system
15) micro:bit connector
14) system access port
1) robot mounting holes
2) micro:bit mounting holes
3) power jack
4) piezospeaker port
5) P16-P17 ports (5V)
6) P18-P19 ports with 5V/VIN select jumper
7) power fault light
8) 3.3V and 5V access
9) breadboard
10) GND, D/A, AD access
11) P20 and P21 LEDs
12) I/O pin access
13) power switch
cyber:bot board
parallax.com/cyberbot
NICERC
6-24 VDC
PWR
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15
P16 P17
P18 P19
VIN
3.3V
5V
GND
P20 P21


Propeller microcontroller I/O pins interact with student's breadboard circuits, piezo speaker, and servo motors
Student writes microPython code in Mu editor for the micro:bit
Propeller carries out requests made by micro:bit code, can reply with data
USB cable
I2C bus
16 17
18 19
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
micro:bit
Propeller sends voltage to D/A output sockets
0 1
D/A
0 1 2
A/D
A/D input sockets send variable voltage input to the micro:bit





Navigation

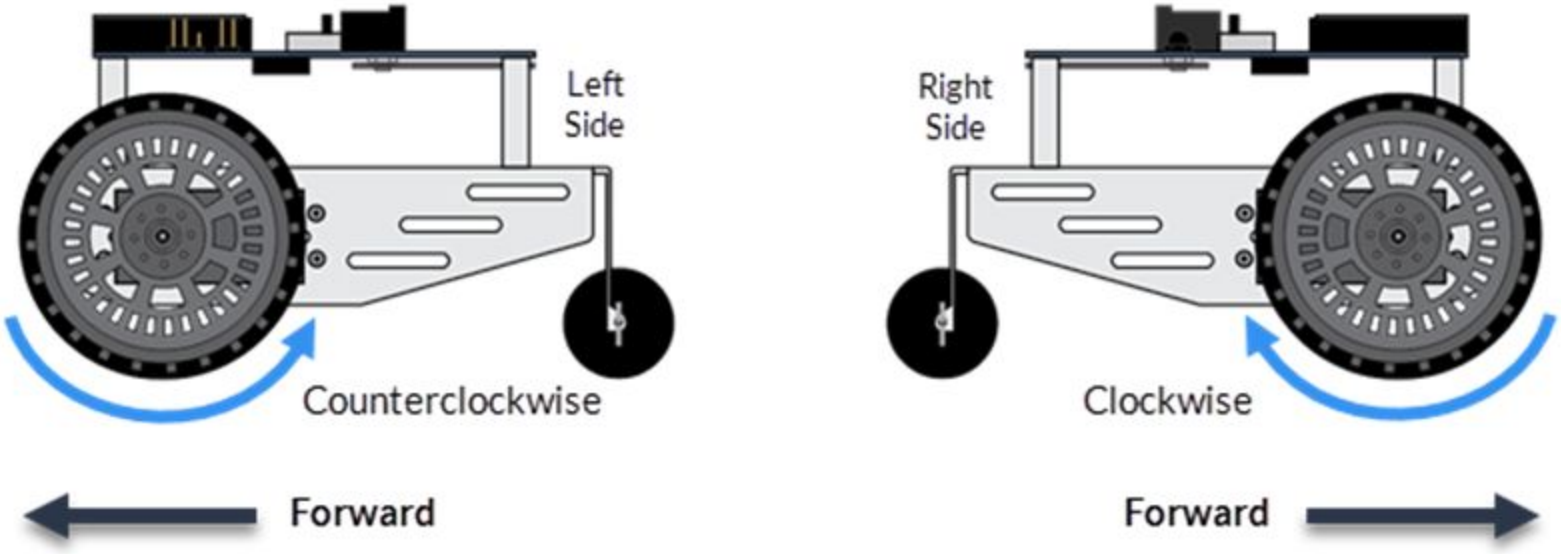
Center the Servos

```
# servo_centering
from cyberbot import *

bot(18).servo_speed(0)
bot(19).servo_speed(0)
```

- Download above script
- cyber:bot switch in position 2
- Turn both potentiometers with screwdriver until servos stop turning

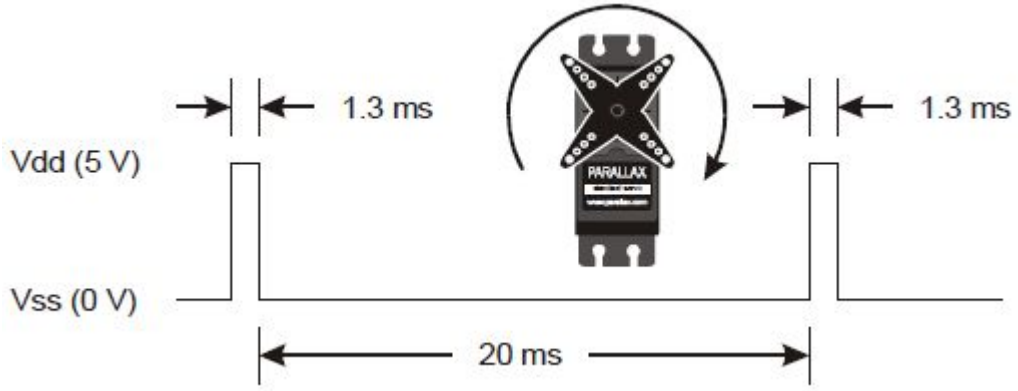
Forward Motion

Means One Servo Turns Opposite Direction



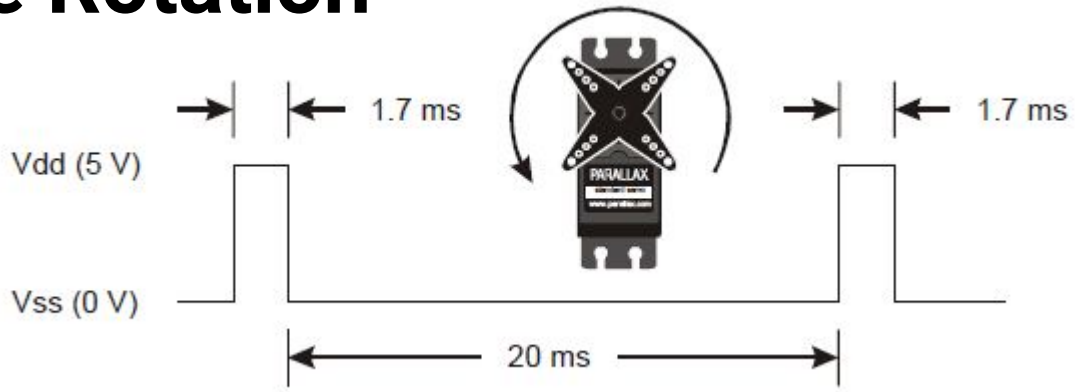
Clockwise Rotation

```
# left_servo_CW.py
from cyberbot import *

bot(18).servo_speed(-75)
```

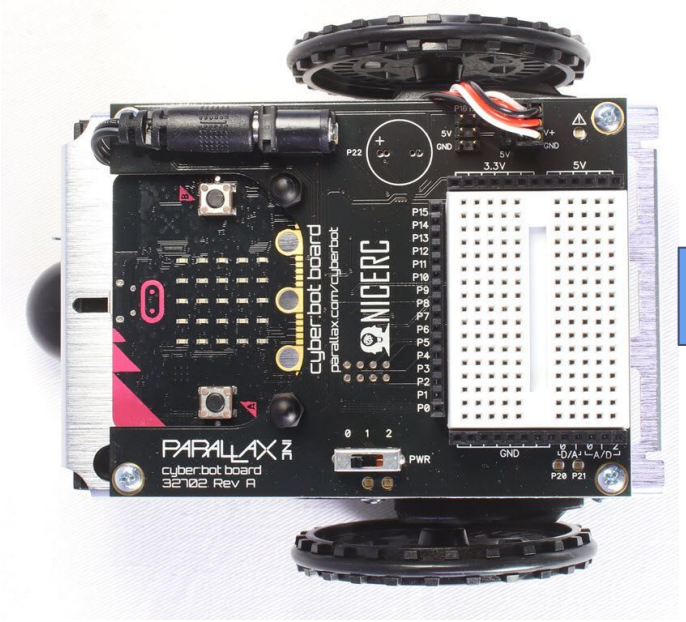


Counter-clockwise Rotation

```
# left_servo_CCW.py
from cyberbot import *

bot(18).servo_speed(75)
```

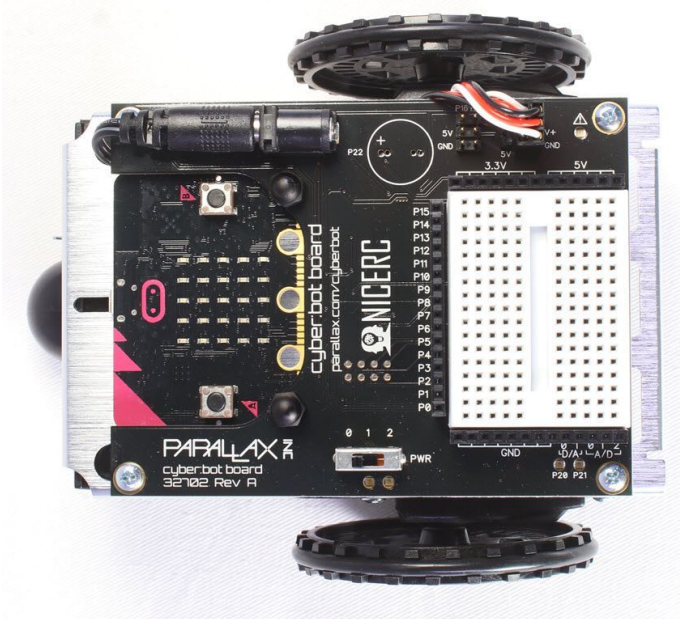


Forward



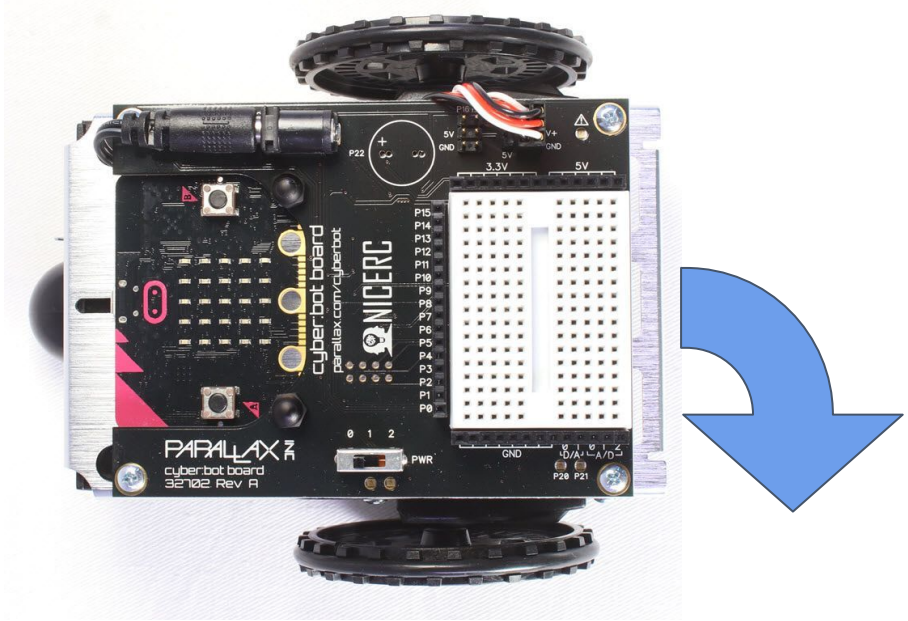

```
from cyberbot import *

# forward_three_seconds.py

bot(18).servo_speed(75)
bot(19).servo_speed(-75)
sleep (3000)

# stop
bot(18).servo_speed(0)
bot(19).servo_speed(0)
```

Backward



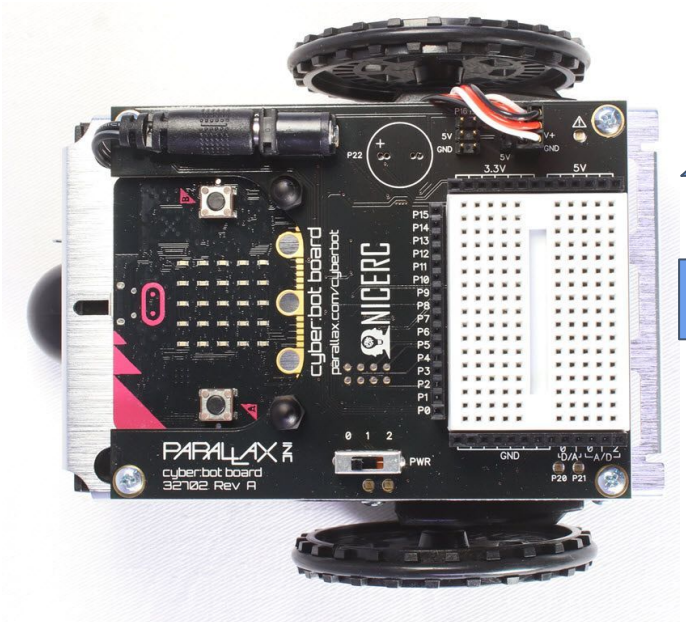

```
from cyberbot import *

# backward_three_seconds.py

bot(18).servo_speed(-75)
bot(19).servo_speed(75)
sleep (3000)

# stop
bot(18).servo_speed(0)
bot(19).servo_speed(0)
```

Right




```
from cyberbot import *

# right_three_seconds.py

bot(18).servo_speed(75)
bot(19).servo_speed(0)
sleep (3000)

# stop
bot(18).servo_speed(0)
bot(19).servo_speed(0)
```

Left



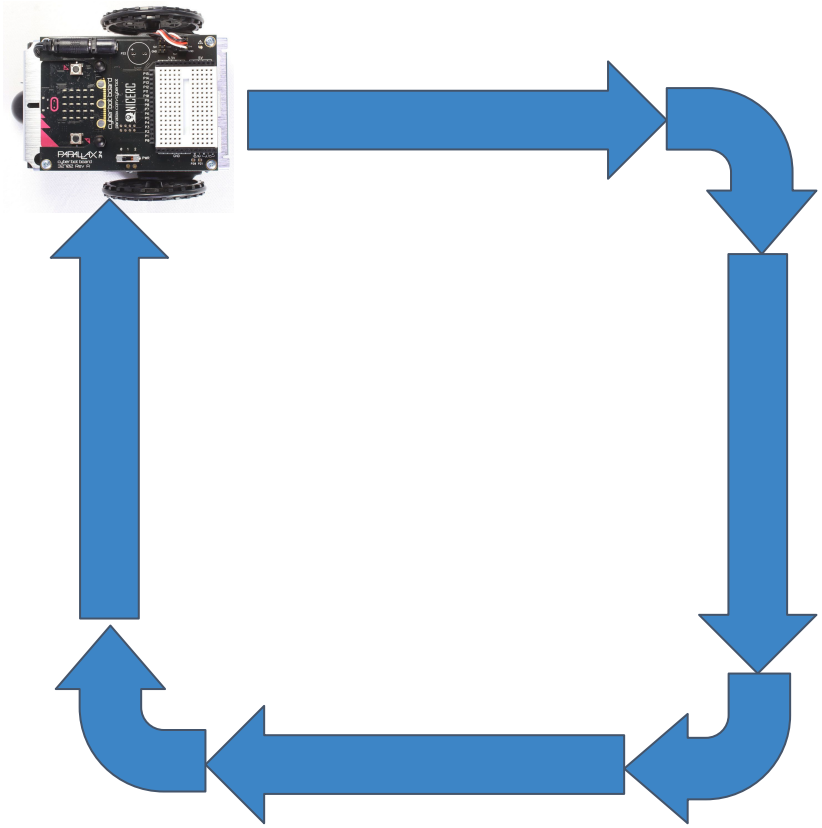

```
from cyberbot import *

# left_three_seconds.py

bot(18).servo_speed(0)
bot(19).servo_speed(-75)
sleep (3000)

# stop
bot(18).servo_speed(0)
bot(19).servo_speed(0)
```

All Together Now

	Left Motor bot(18).servo_speed	Right Motor bot(19).servo_speed
Forward	75	-75
Backward	-75	75
Left	0	-75
Right	75	0

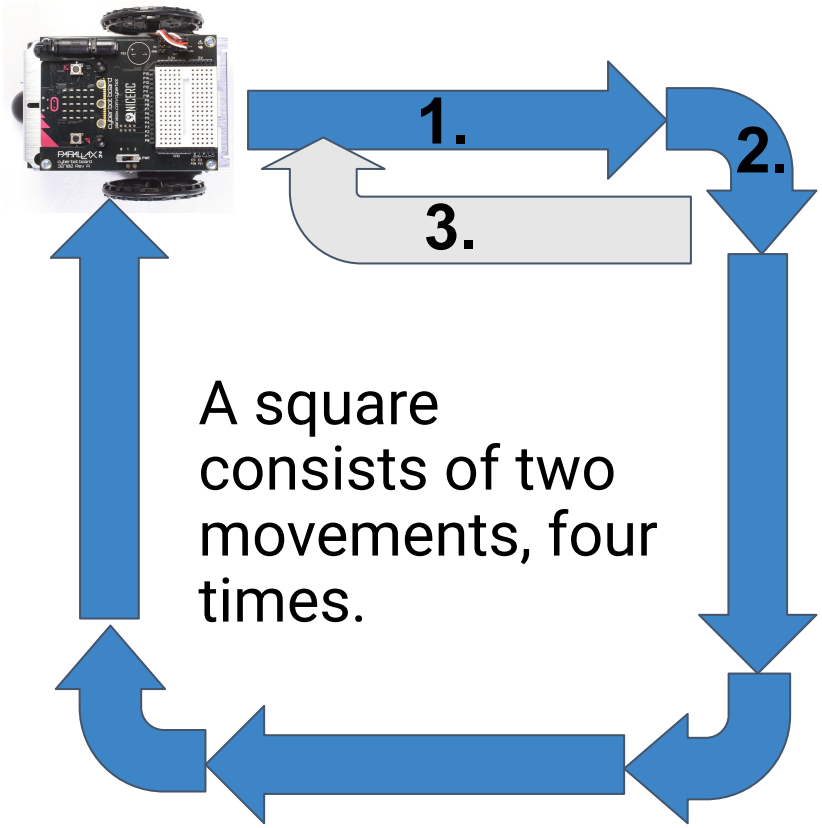
Square

Write a Python script for your cyber:bot



```
from cyberbot import *
# square.py

# straight
bot(18).servo_speed(75)
bot(19).servo_speed(-75)
sleep (3000)

# right
bot(18).servo_speed(75)
bot(19).servo_speed(0)
sleep (2000)

# you write the rest
# of the script
```

Repeat Loops




```
from cyberbot import *
# square_with_repeat

for y in range (0, 3):
    # straight
    bot(18).servo_speed(75)
    bot(19).servo_speed(-75)
    sleep (3000)

    # right
    bot(18).servo_speed(75)
    bot(19).servo_speed(0)
    sleep (2000)
```

Functions Without Arguments

- Simplified the drive commands
- Functions “lock in” the speed and duration
- Why not pass arguments of speed and duration?

```
from cyberbot import *
# functions without arguments

def straight():
    bot(18).servo_speed(75)
    bot(19).servo_speed(-75)
    sleep (3000)

def right():
    bot(18).servo_speed(75)
    bot(19).servo_speed(0)
    sleep (1100)

def stop():
    bot(18).servo_speed(0)
    bot(19).servo_speed(0)

straight()
right()
straight()
right()
stop()
```

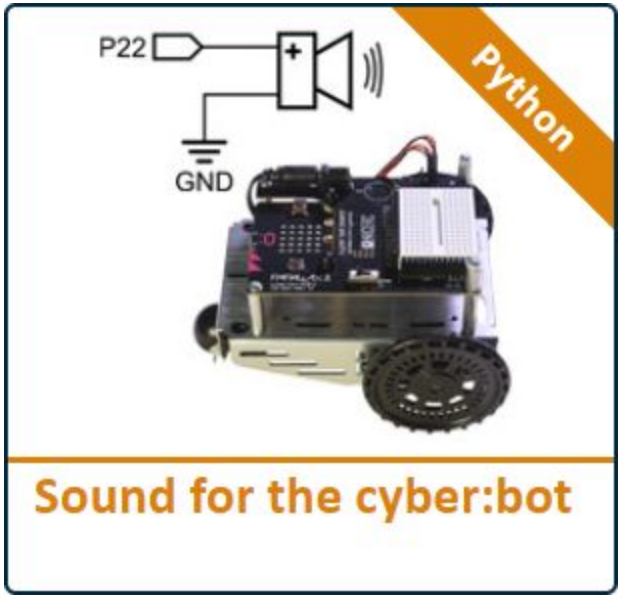
Functions with Arguments

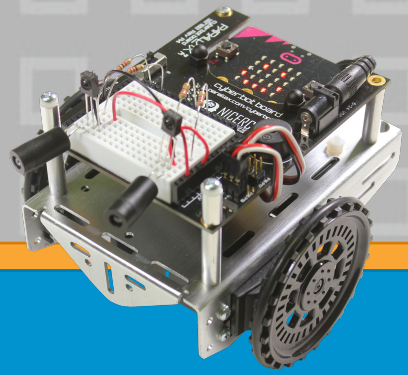
- Use the functions and pass values as arguments
- Creates simplified code; easier to read

- Write Python code and use functions to draw a triangle.

```
from cyberbot import *
# functions with arguments

def straight(duration):
    bot(18).servo_speed(25)
    bot(19).servo_speed(-25)
    sleep (dur)

def right(duration):
    bot(18).servo_speed(25)
    bot(19).servo_speed(0)
    sleep (duration)

def stop(duration):
    bot(18).servo_speed(0)
    bot(19).servo_speed(0)
    sleep(duration)

straight(1000)
right(500)
straight(1000)
right(500)
stop(0)
```

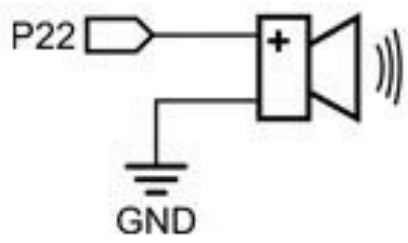
Functions with Loops



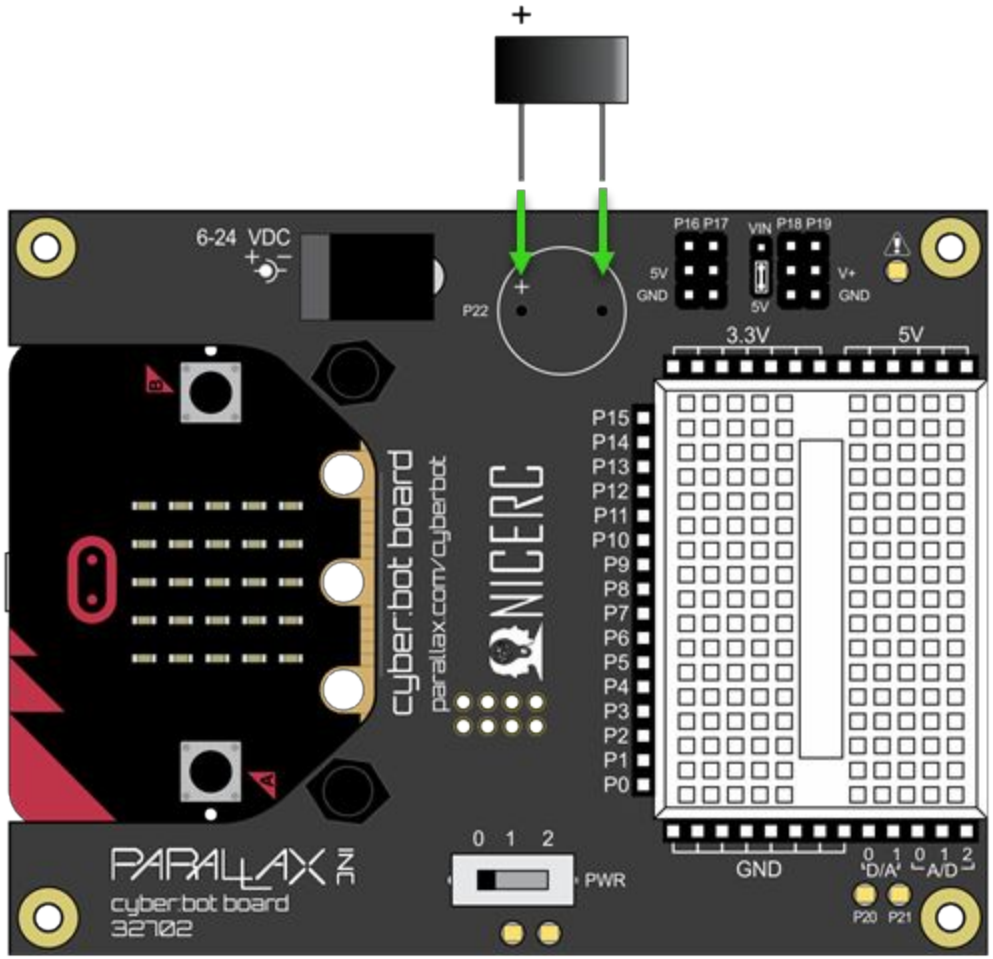


Sounds

Place the Piezospeaker



- Find piezospeaker in the Small Robot Electronics Component Pack.
- Peel off the “Remove the seal after washing” sticker if it has one.
- Plug into the cyber:bot board



The Tone Command

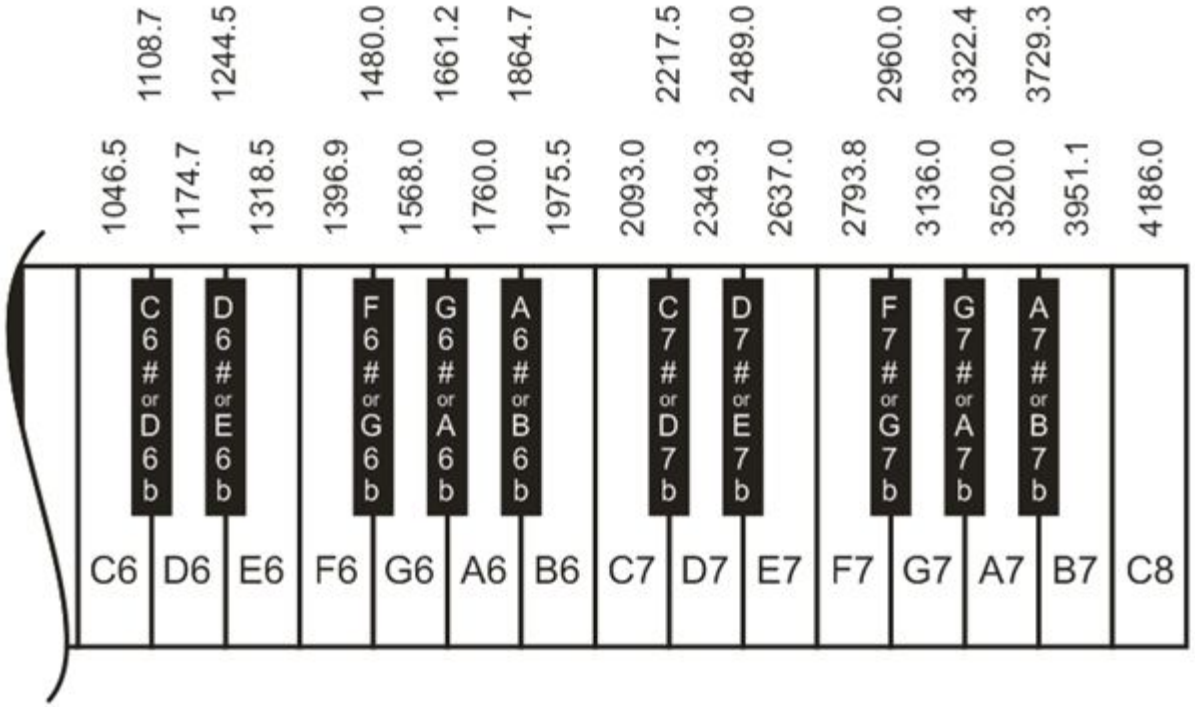
- Sounds are used in robotics for program feedback (sensor actions), customizing behavior, or as alarms.
- Syntax for the tone command:
  - bot(22).tone(frequency, milliseconds)

- What will this script do? Hint, the loop counter has a step value!

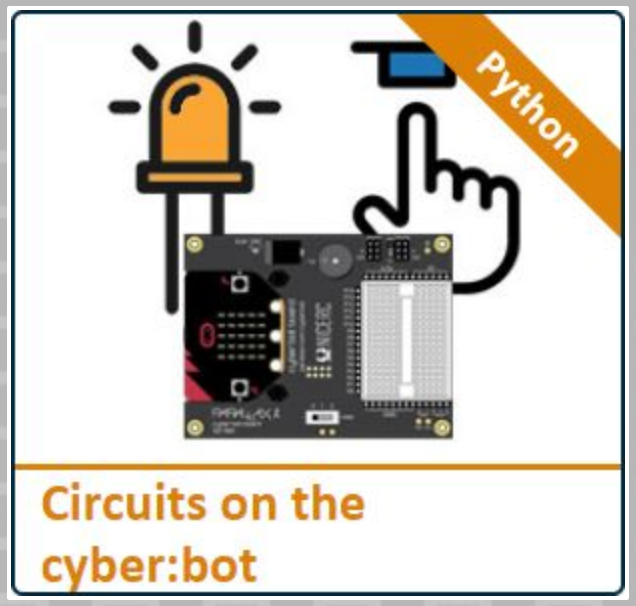
```
# sound_effect.py

from cyberbot import *

for freq in range (500, 3100, 100):
        bot(22).tone(freq, 100)
```

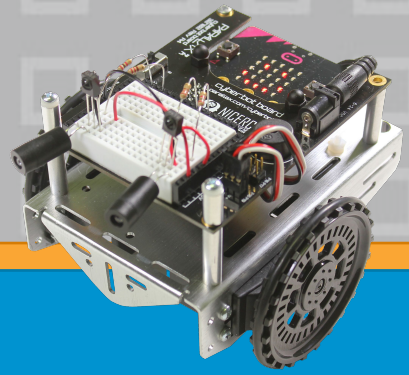
Notes from Frequencies

Happy Birthday!




```
happy_birthday.py

from cyberbot import *

bot(22).tone(2349.3, 125)          #D for an eighth
bot(22).tone(2349.3,  62)          #D for a sixteenth
bot(22).tone(2637.0, 250)          #E for a quarter
bot(22).tone(2349.3, 250)          #C for a quarter
bot(22).tone(3136.0, 250)          #G for a quarter
bot(22).tone(2793.8, 500)          #F for a half
```



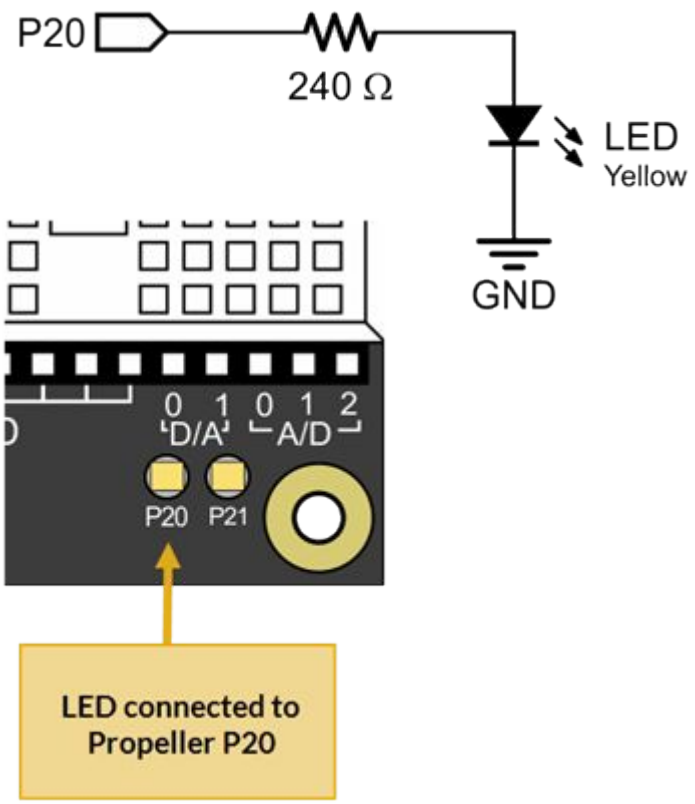


Circuits

Blink a Light: Using Built-in LEDs

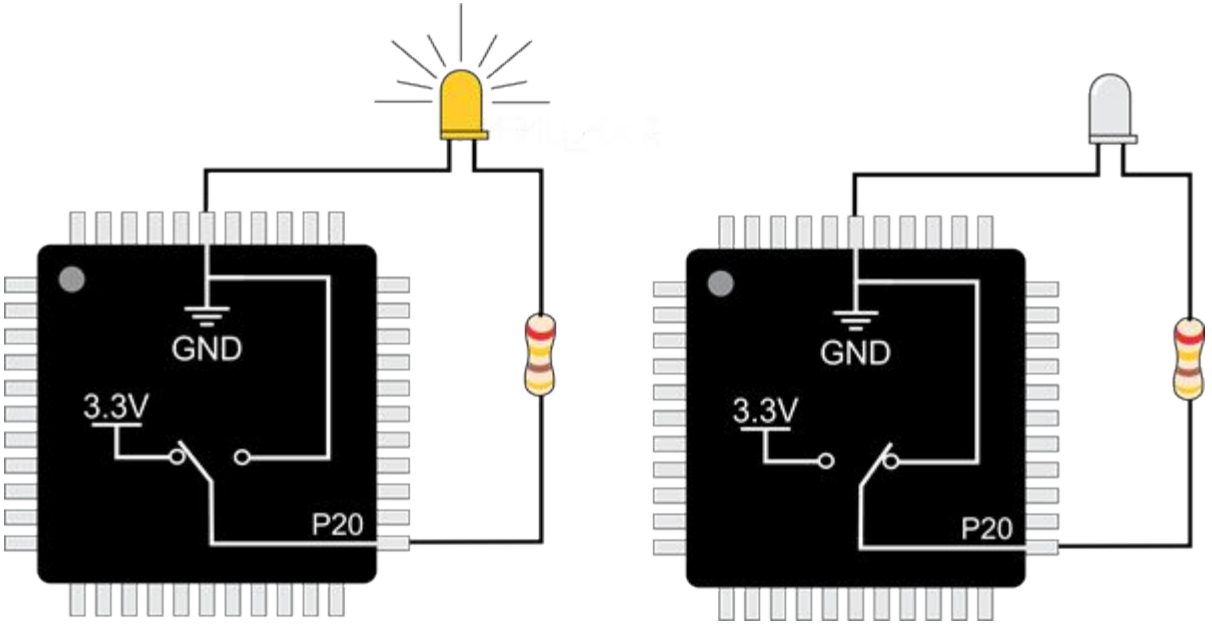
- Pins 20 and 21 each have an LED connected directly.
- Modify the code to alternate with P21.
- Speed it up!

```
# pin_20_blink.py

from cyberbot import *

while True:
    bot(20).write_digital(1)
    sleep(2000)
    bot(20).write_digital(0)
    sleep(1000)
```

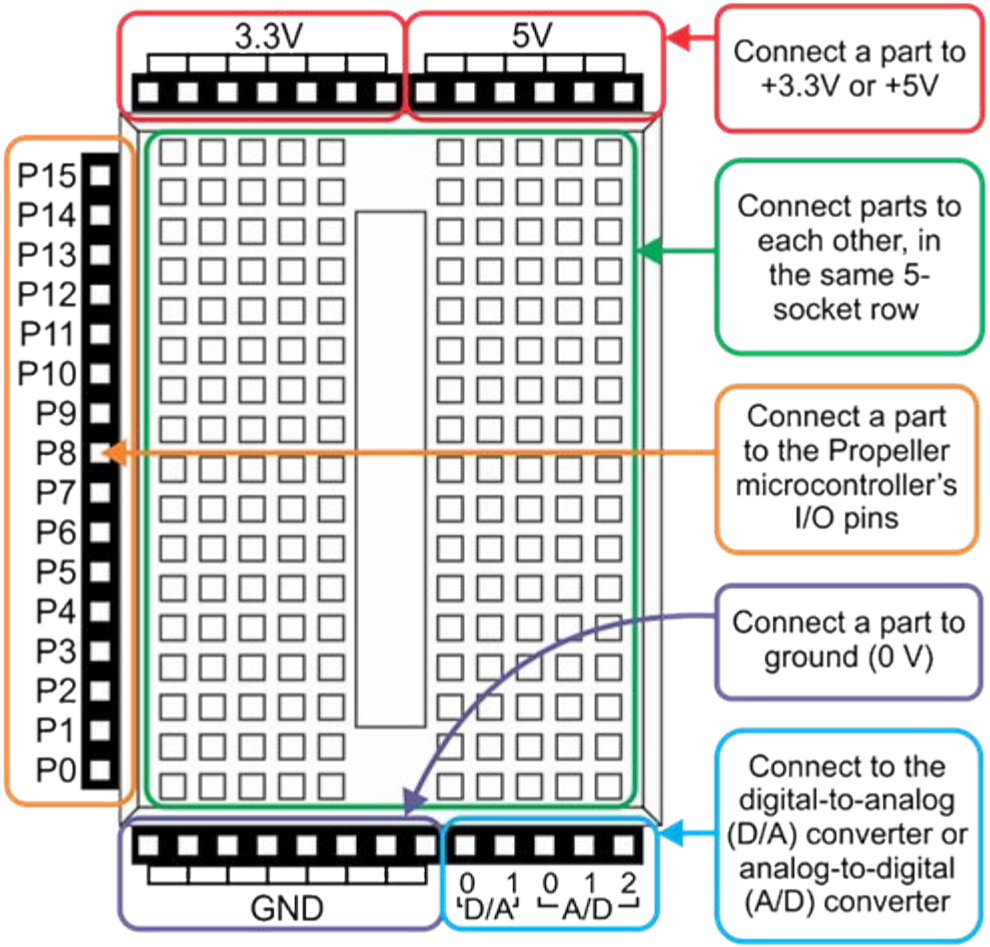


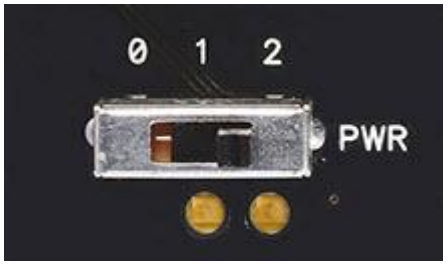
Blink a Light: How it Works

- `bot(20).write_digital(1)` sets the pin to “output high” and connects to the 3.3V power supply.
- `bot(20).write_digital(0)` sets the pin to “output low” and connects the pin to ground.



Blink a Light: Breadboard Basics

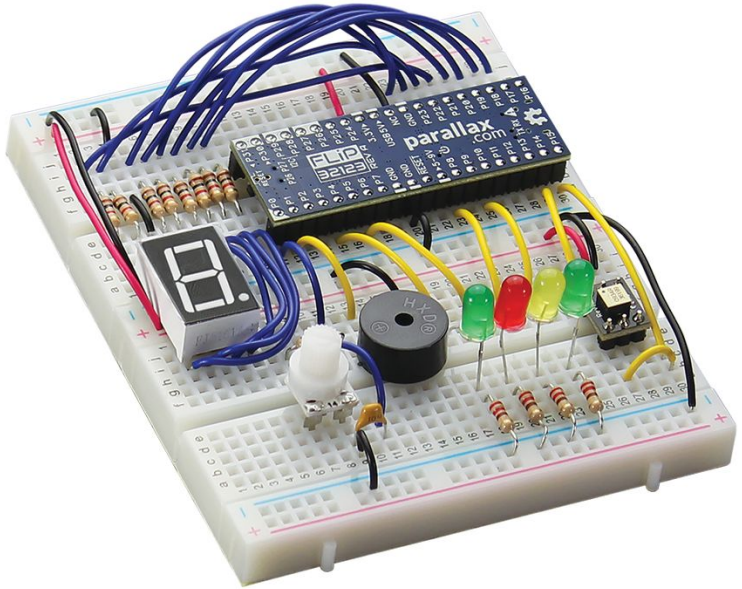




- **Position 0:** use for building circuits, flashing scripts.
- **Position 1:** powers breadboard, Propeller.
- **Position 2:** powers breadboard, Propeller, and motors (3-pin ports)

Blink a Light: Breadboard Basics

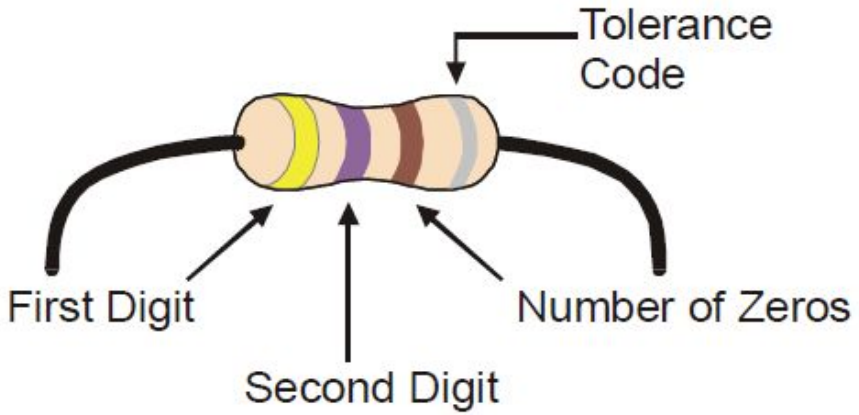
- Electronic projects are easily built on breadboards
- Read schematics and pictorials to learn how to place components
- Prototype, proof of concept or student projects



Blink a Light: Resistors

Digit	Color
0	Black
1	Brown
2	Red
3	Orange
4	Yellow
5	Green
6	Blue
7	Violet
8	Gray
9	White

- **Resistor** is a component that “resists” the flow of electrical current
- **Current** is the flow of electricity
- Each resistor has a value that tells how strongly it resists current flow.



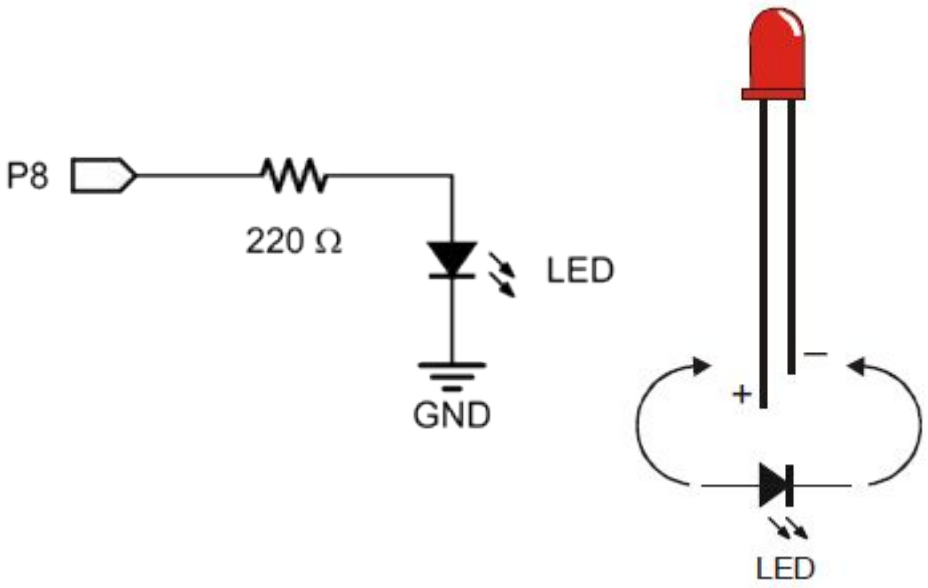
$47 \times 10^1 = 470$

470 Ohm resistor

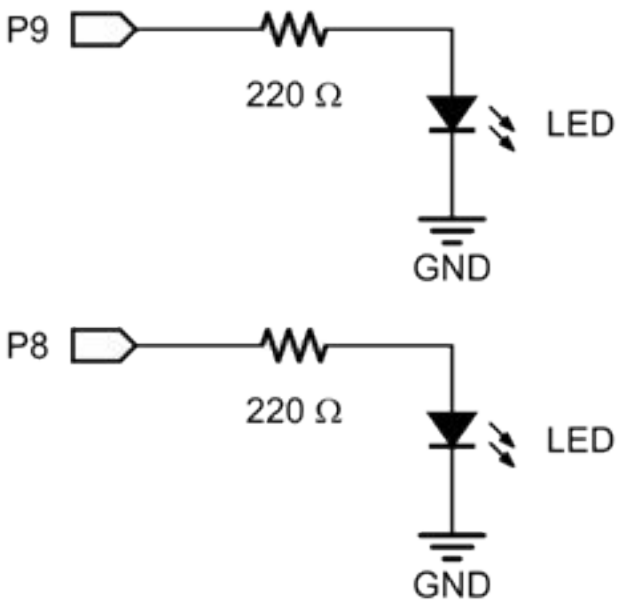
Blink a Light: LED

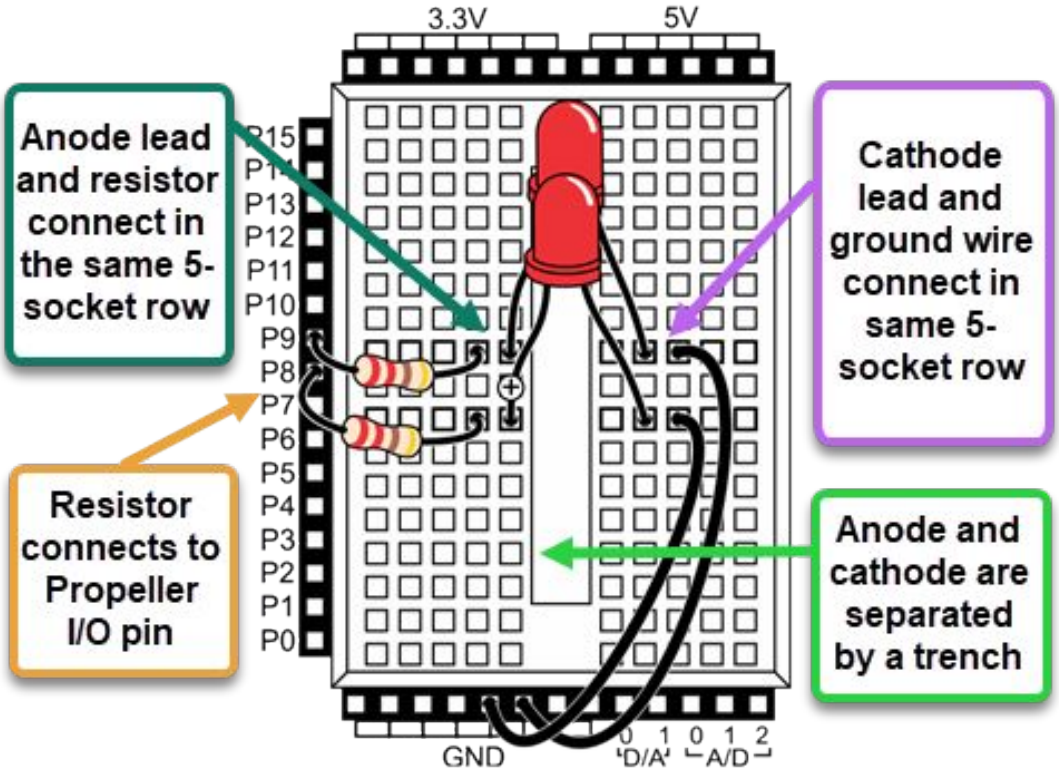
- An LED has two terminals:
  - anode lead is labeled with the plus-sign (+), and it is the wide part of the triangle in the schematic symbol.
  - cathode lead is labeled with a minus-sign (-), and it is the line across the point of the triangle in the schematic symbol

- Light-emitting diode (LED) emits light when current passes through it.



Blink a Light: Build an LED Circuit

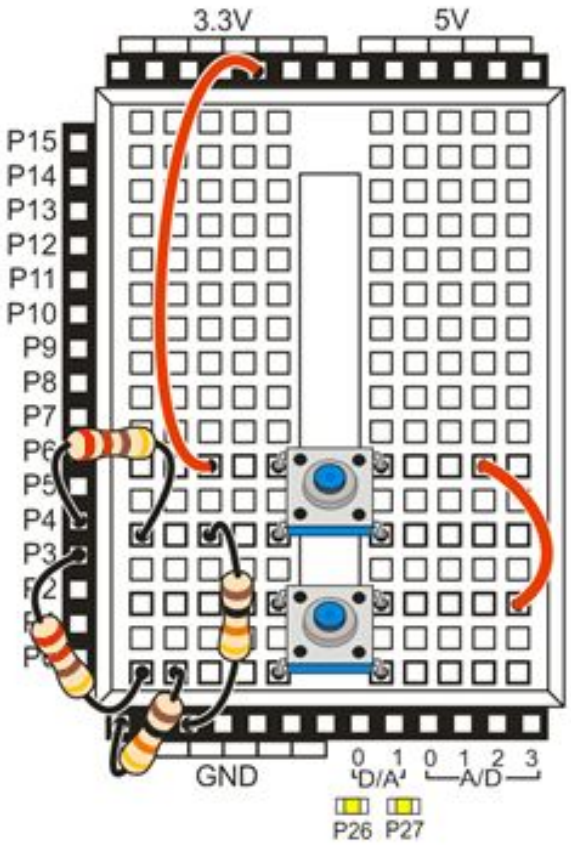


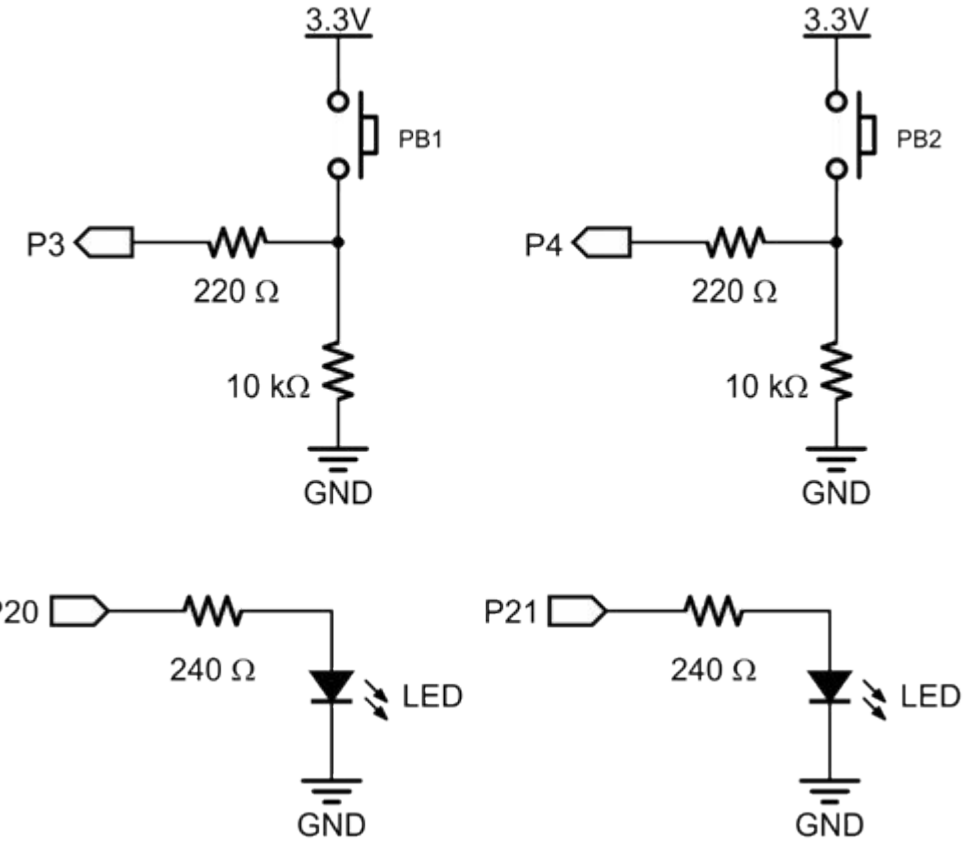


- Write a program to alternate blinks on these two LEDs.

Pushbuttons: Add to the LEDs on Breadboard

- Light LEDs on P20/P21 when buttons pressed



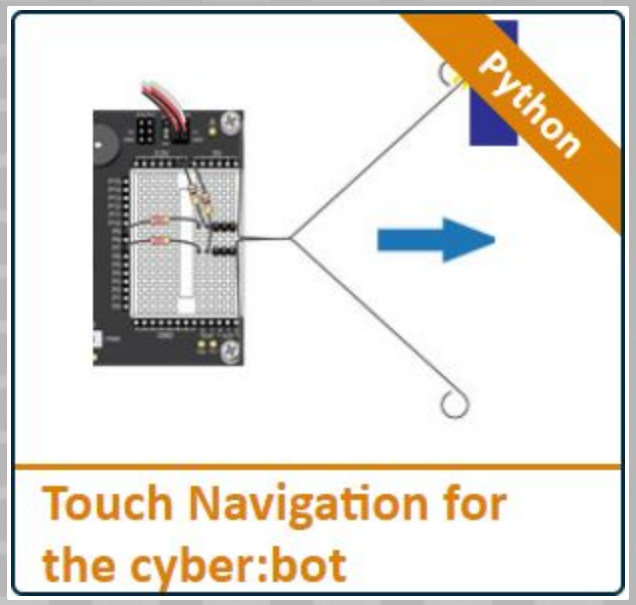


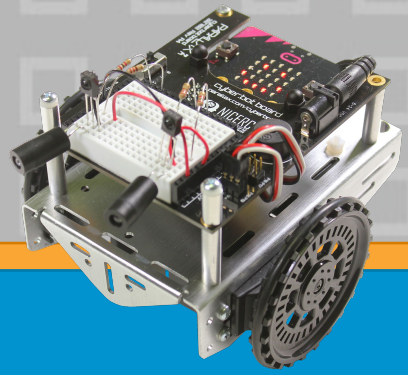
Pushbuttons: Add to the LEDs on Breadboard

- **`if-elif`** conditional statement
- P20 LED is on when button on P3 is pressed

```
# pin_3_button_LED.py
from cyberbot import *

while True:
    if bot(3).read_digital() == 0:
        bot(20).write_digital(0)
    elif bot(3).read_digital() == 1:
        bot(20).write_digital(1)
```

- Modify code to control P21 LED with P4 button.
- Make P3 button control P20 LED and P4 button control P21 LED

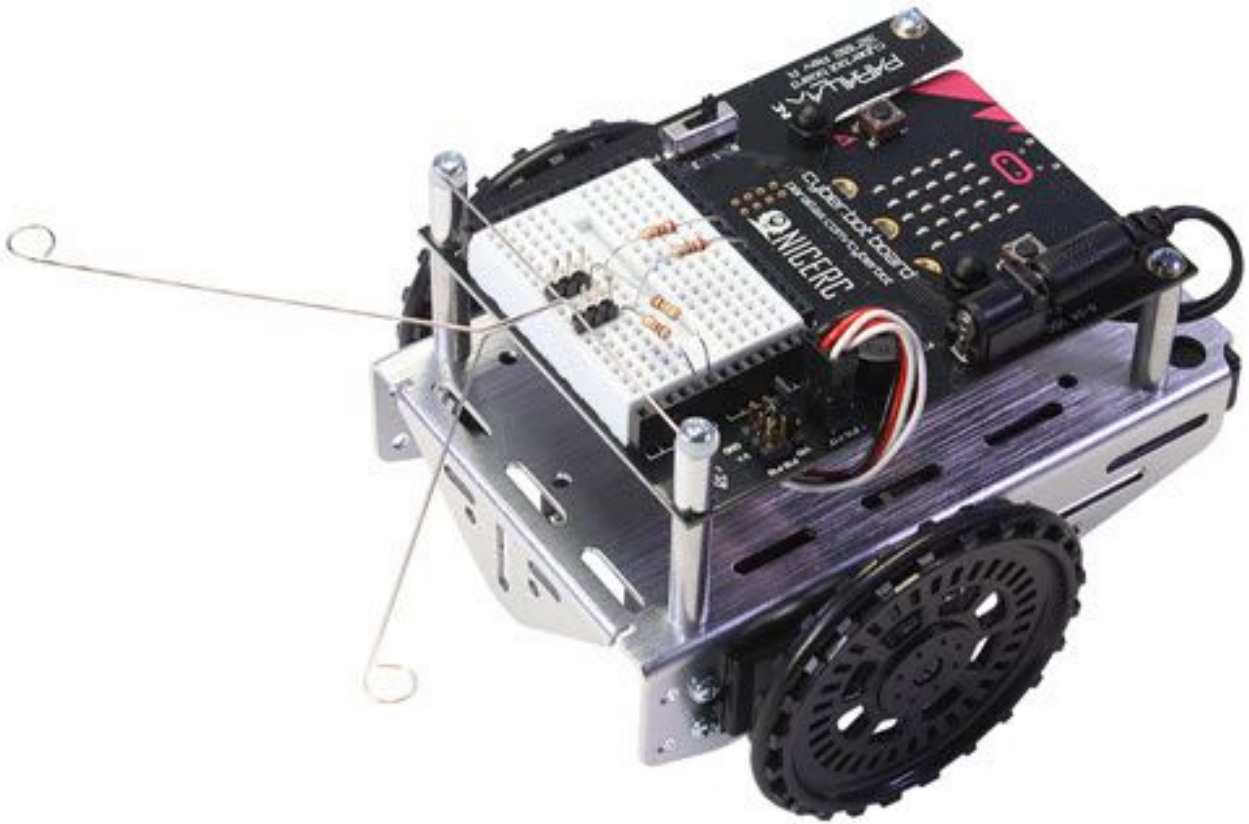




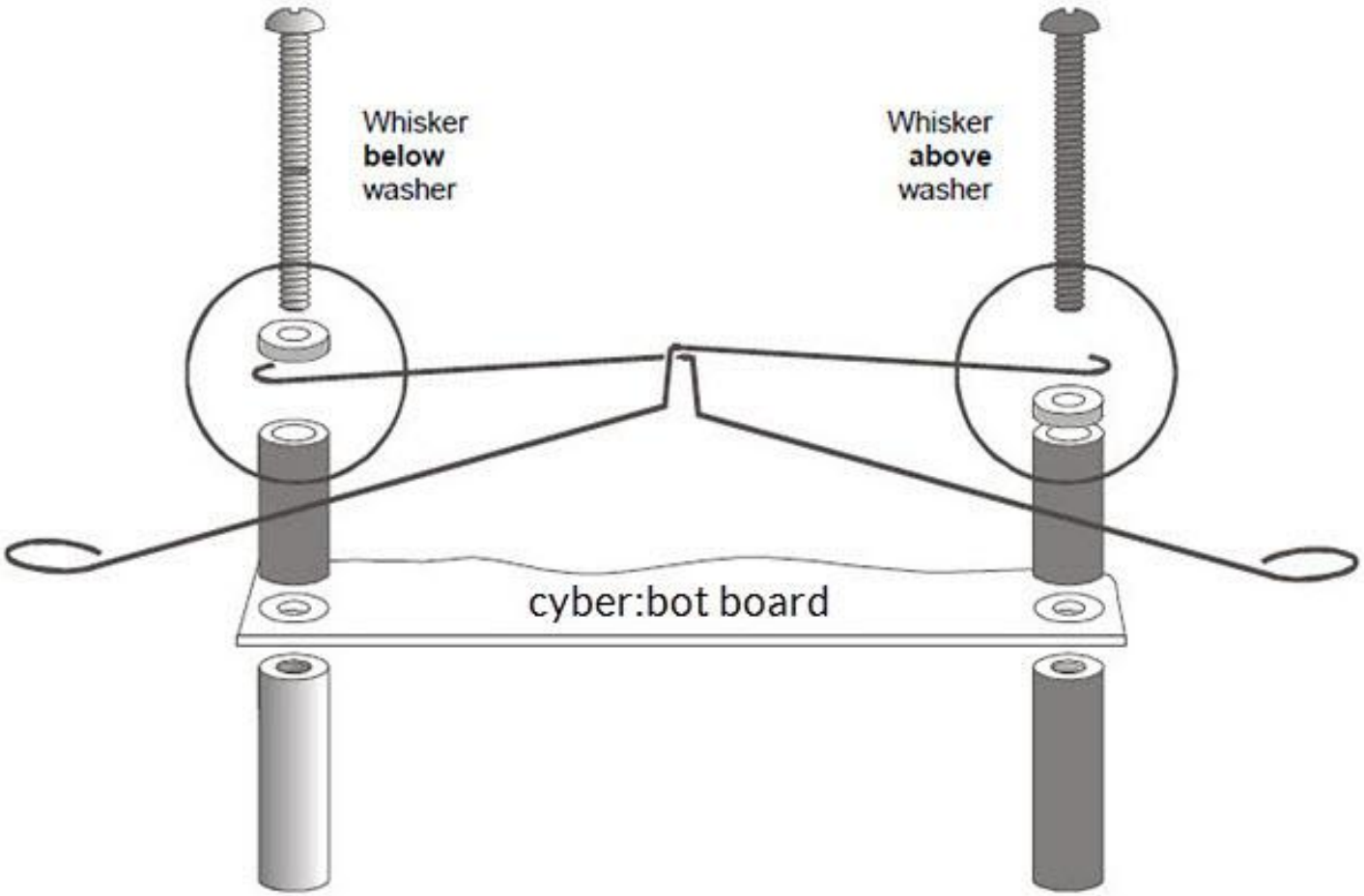
Touch Navigation


cyber:bot
robot kit
NICERC

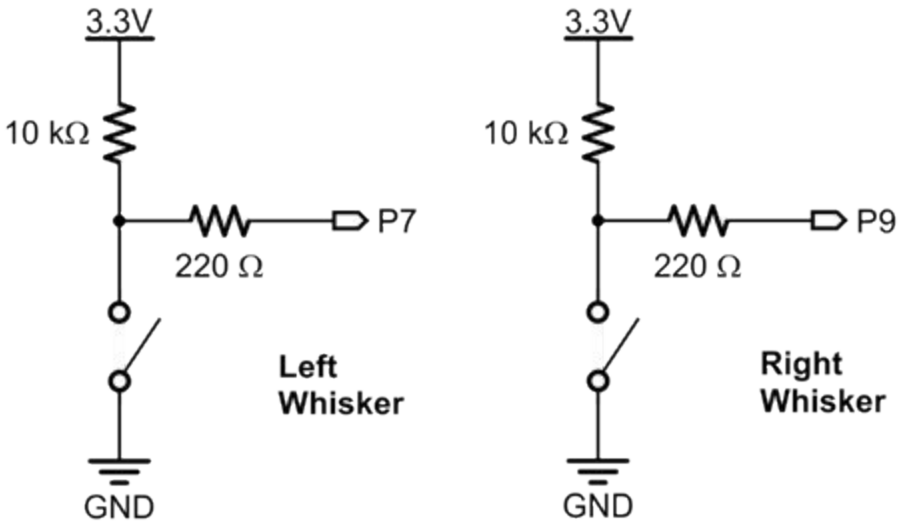
Touch Navigation: Assembled Circuit

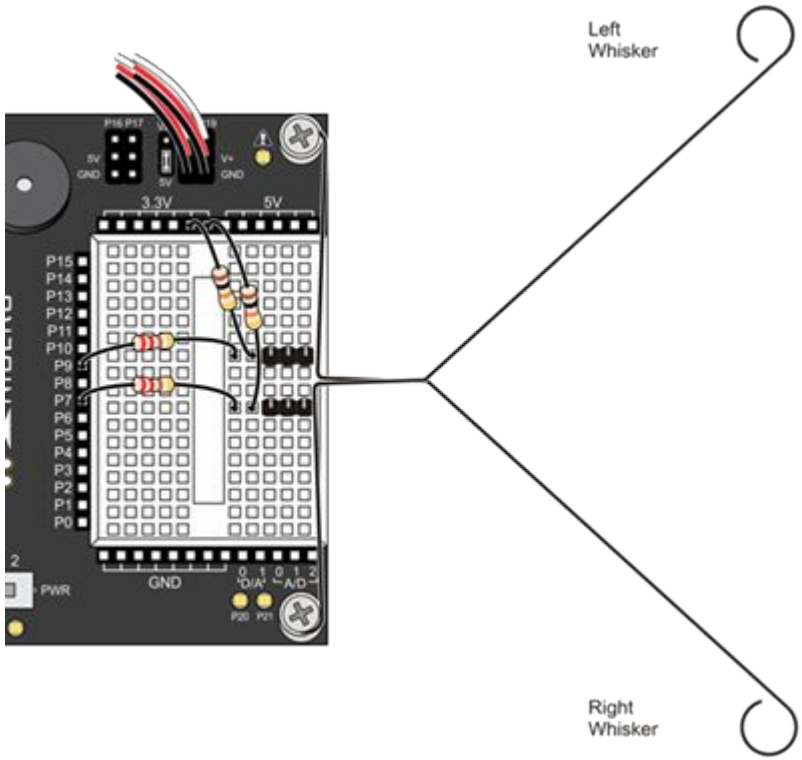


Touch Navigation: Circuit Build



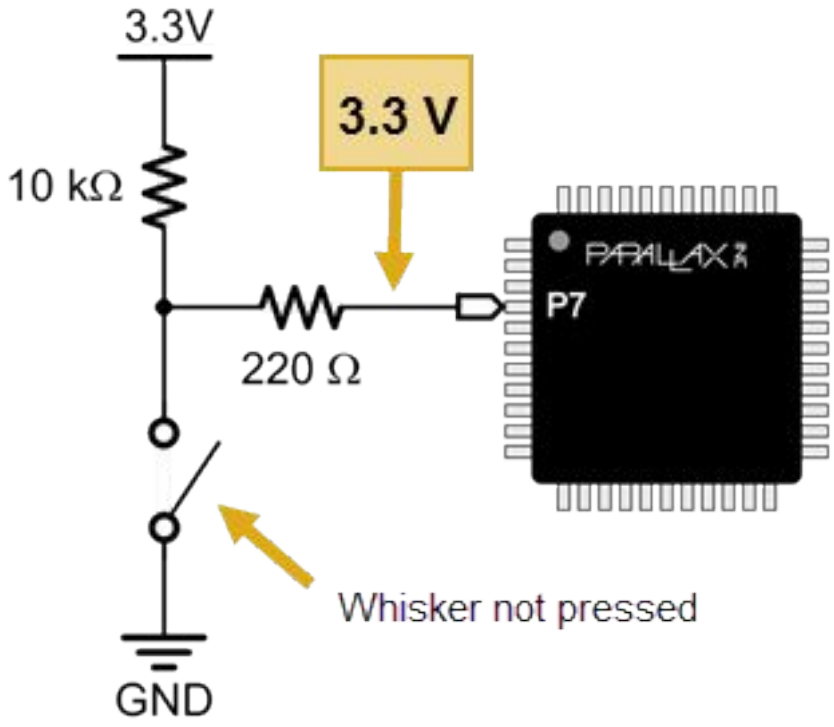
Touch Navigation: Circuit Build



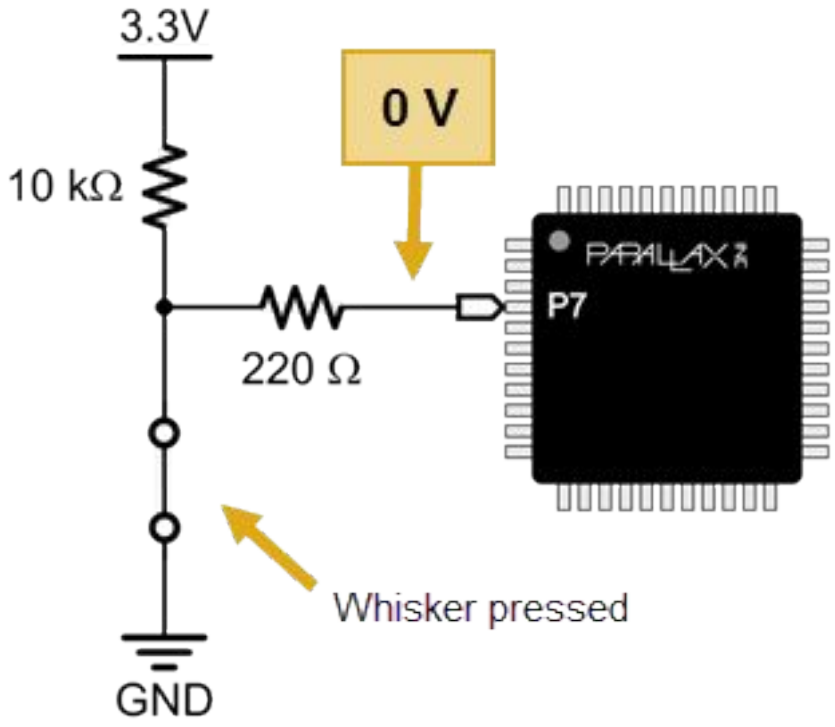


Touch Navigation: Pressed / Not Pressed

`bot(7).read_digital()`
returns **1**

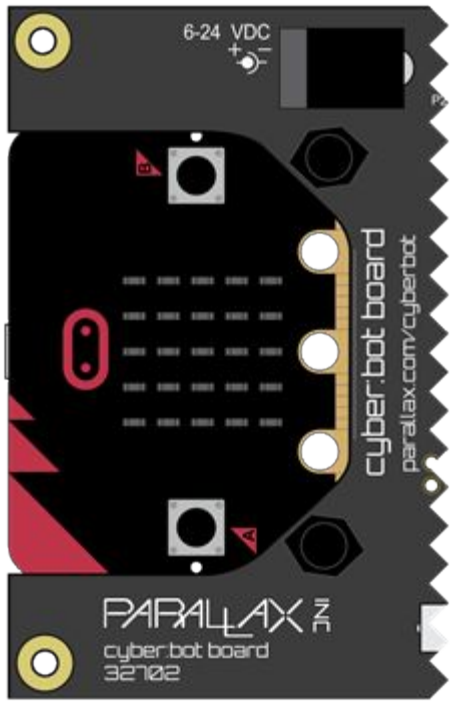


`bot(7).read_digital()`
returns **0**

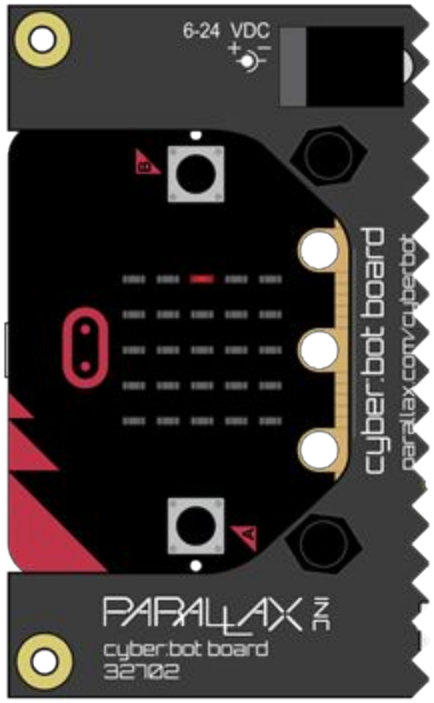


Touch Navigation: Testing Circuit

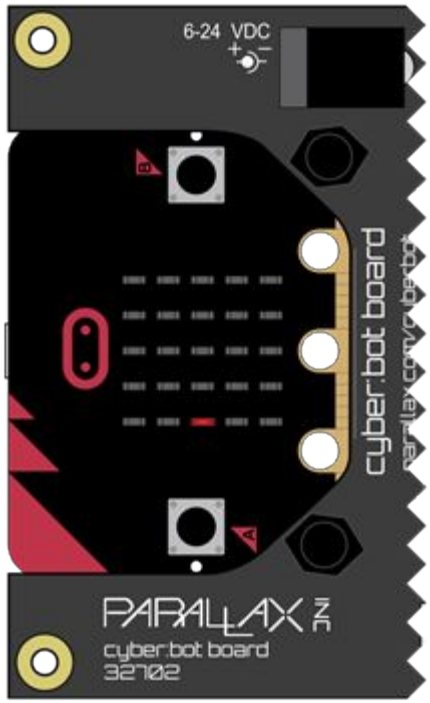
- LEDs show state of whiskers (code from tutorial)



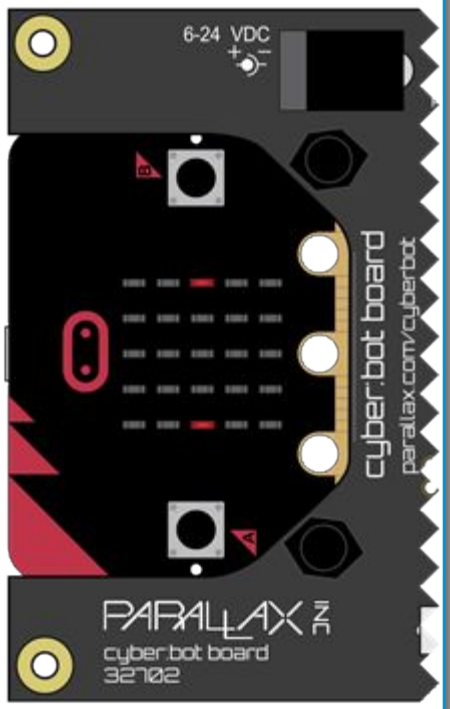
Neither pressed.



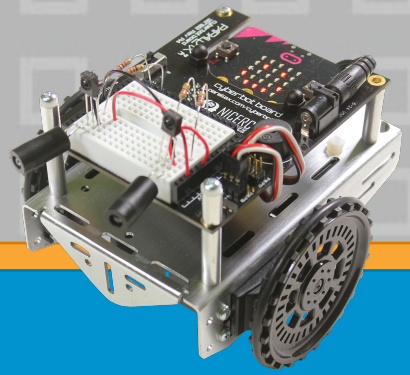
Left pressed.



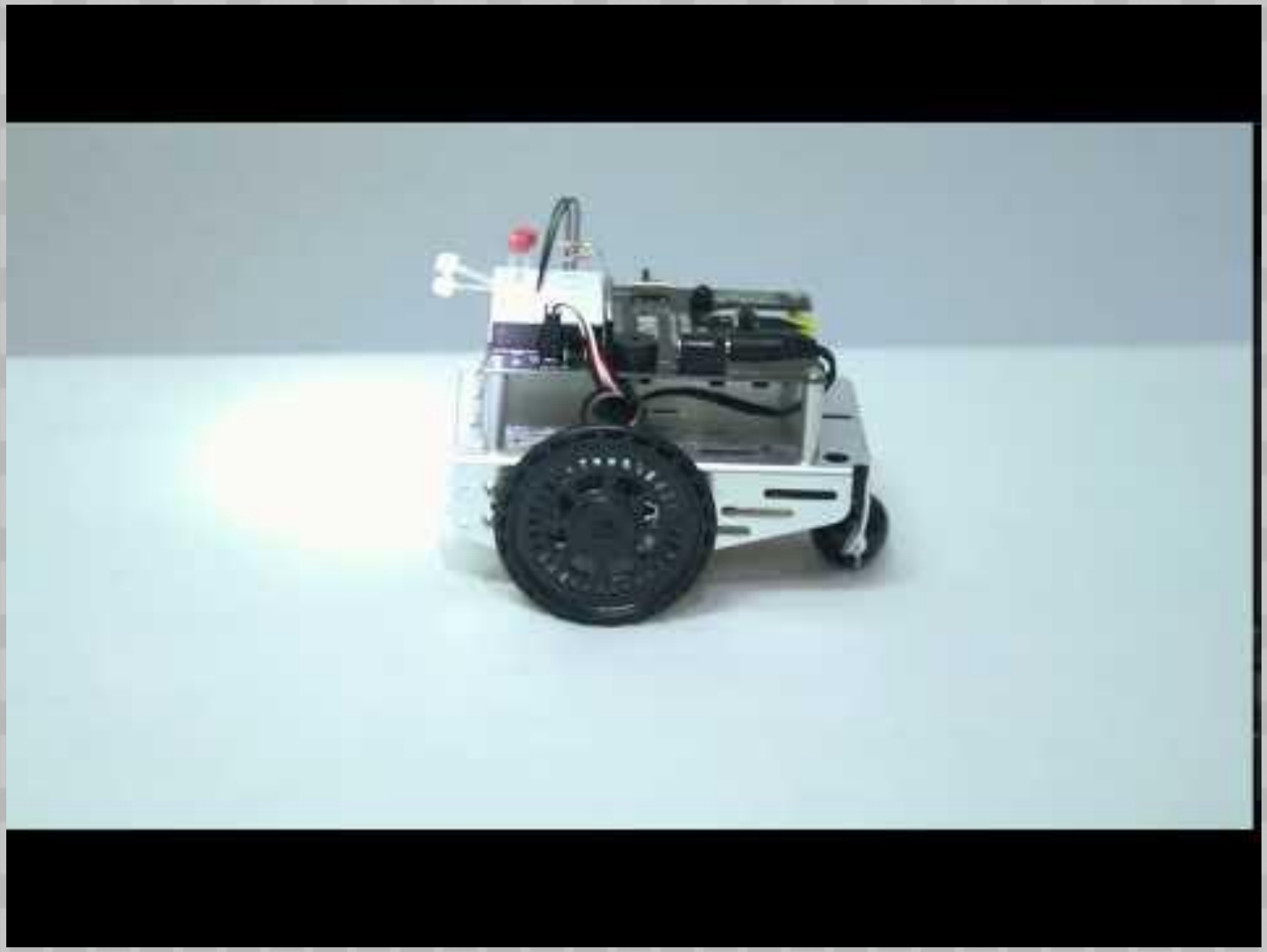
Right pressed.



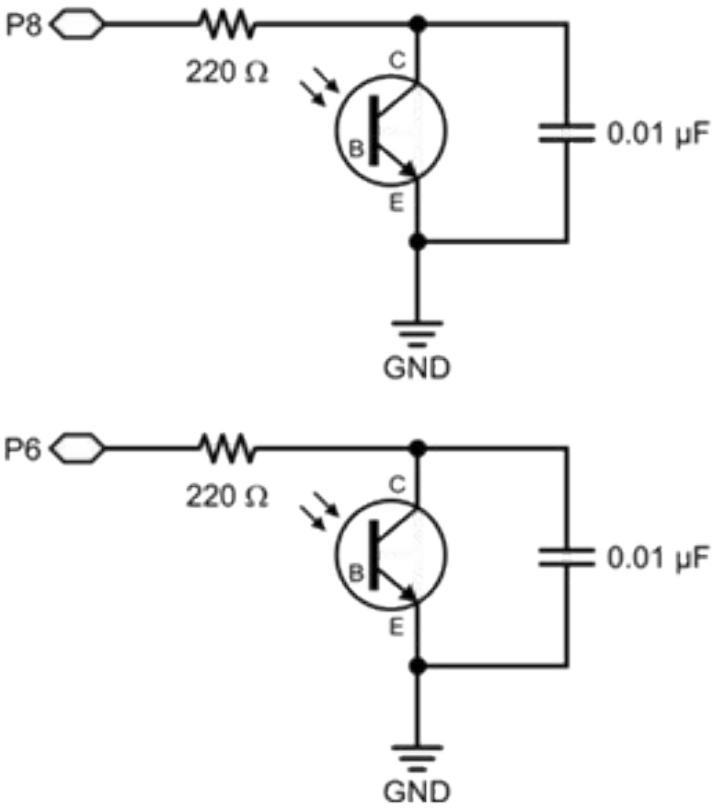
Both pressed.

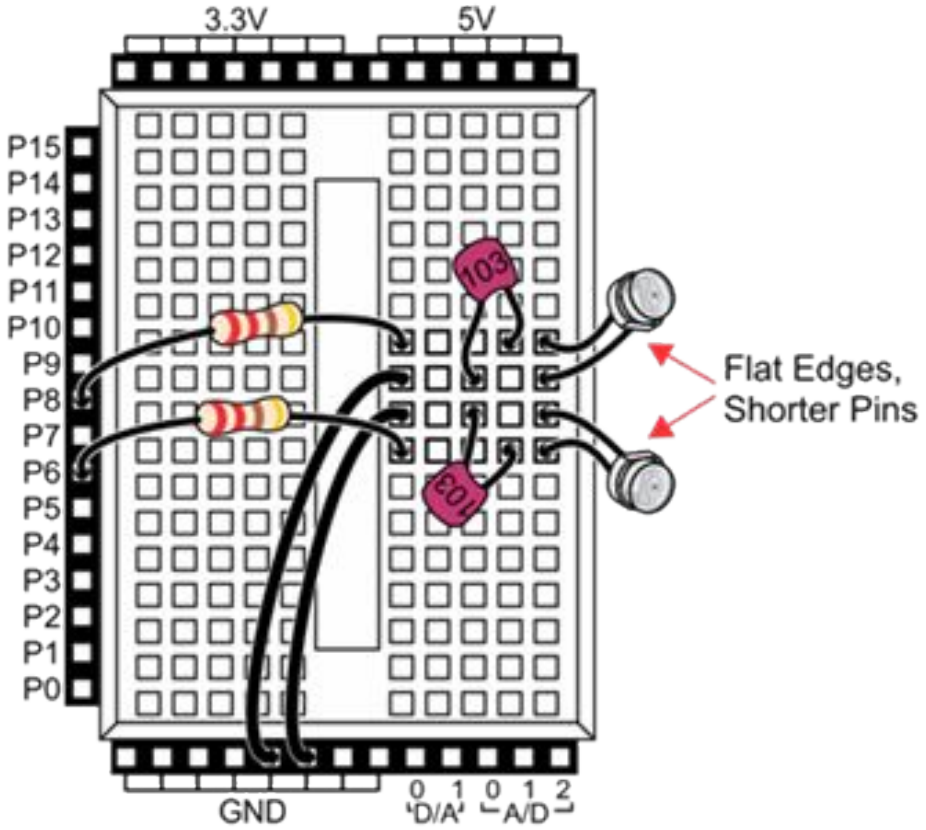


Visible Light

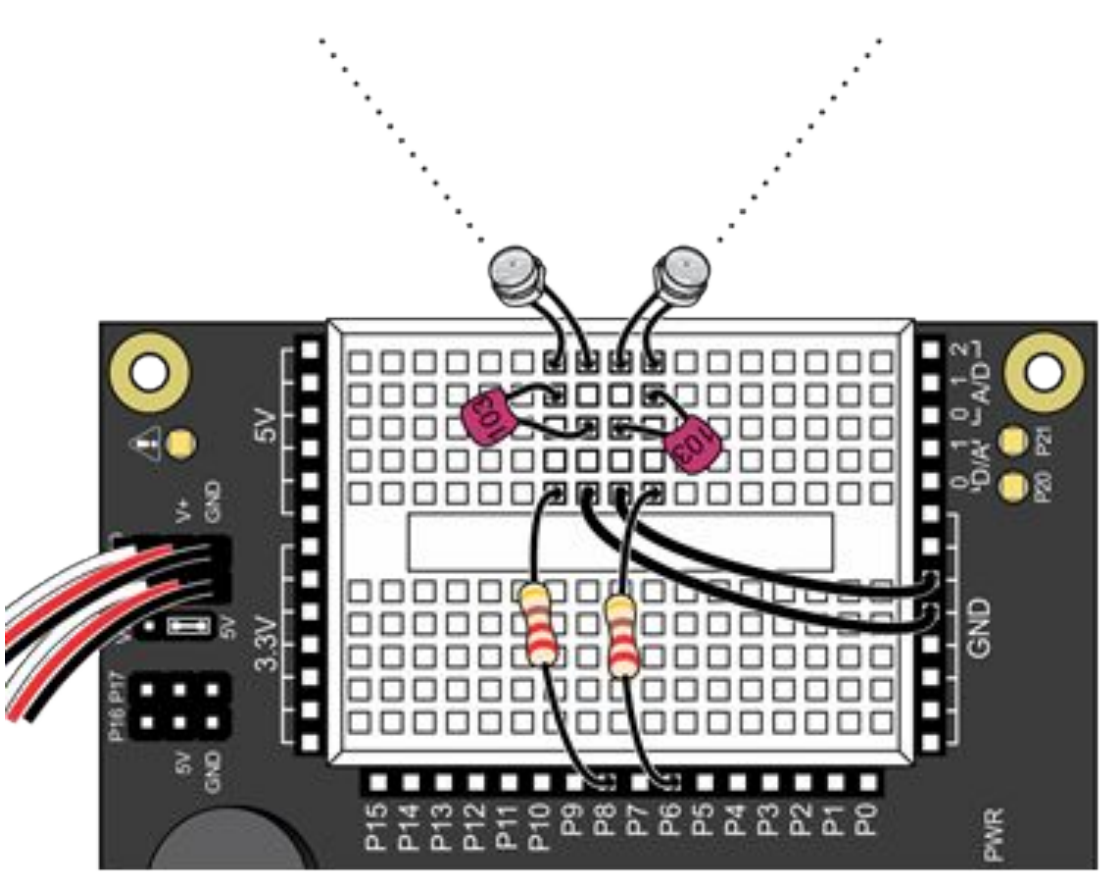


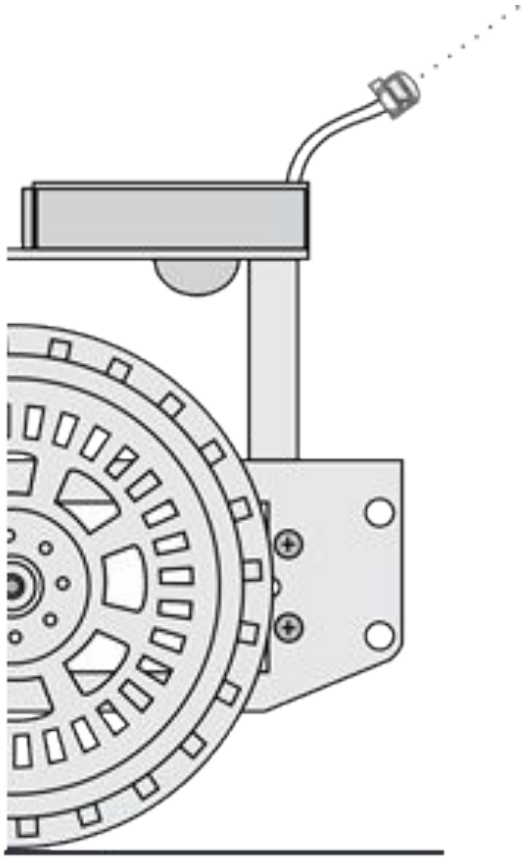
Visible Light Following: Circuit Build

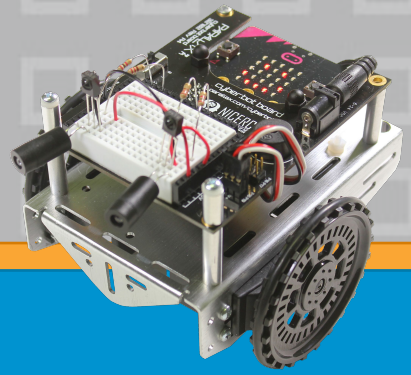




Visible Light Following: Circuit Build



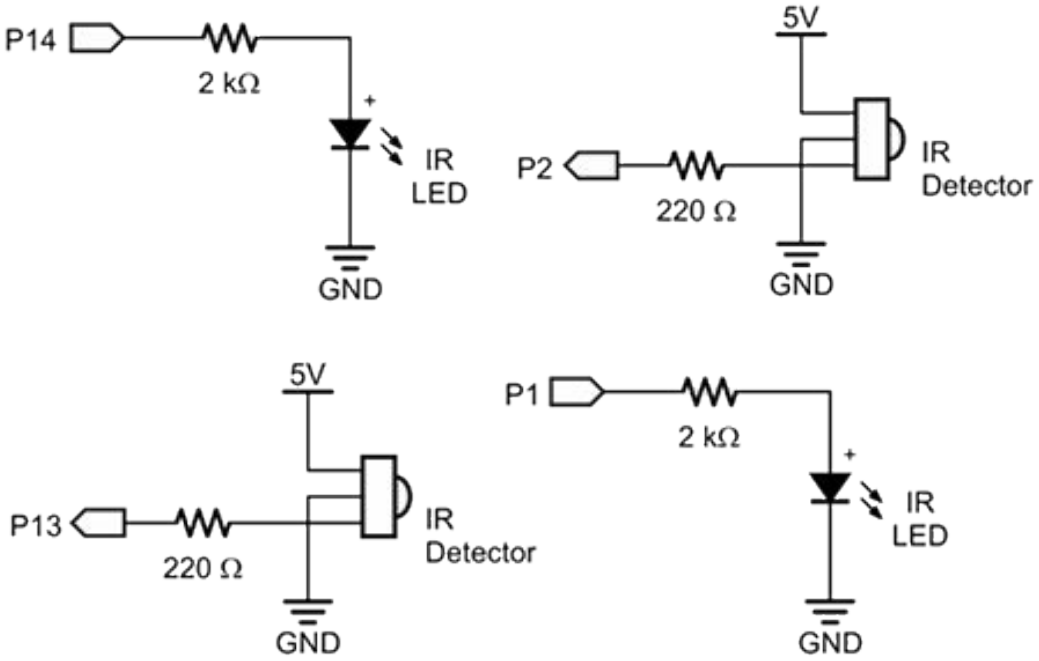


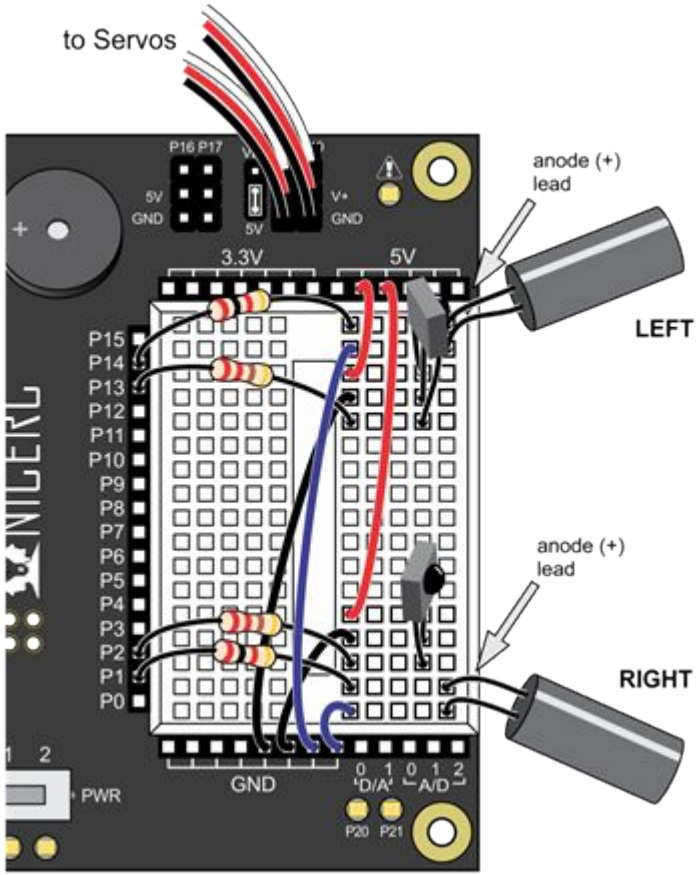


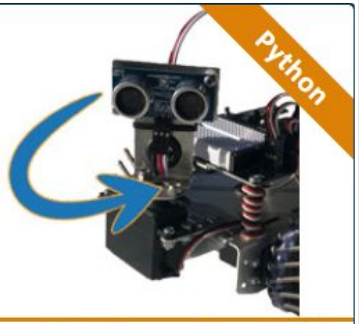
Infrared Light


cyber:bot
robot kit
NIGERC

Infrared Object Avoidance: Circuit Build



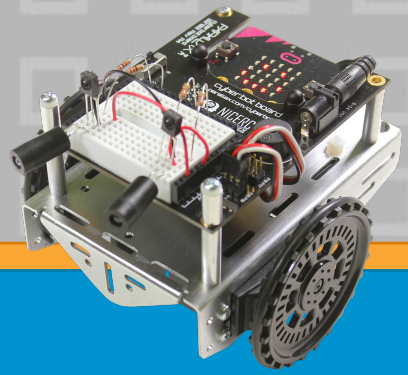




cyber:bot Roaming with the Ping)))

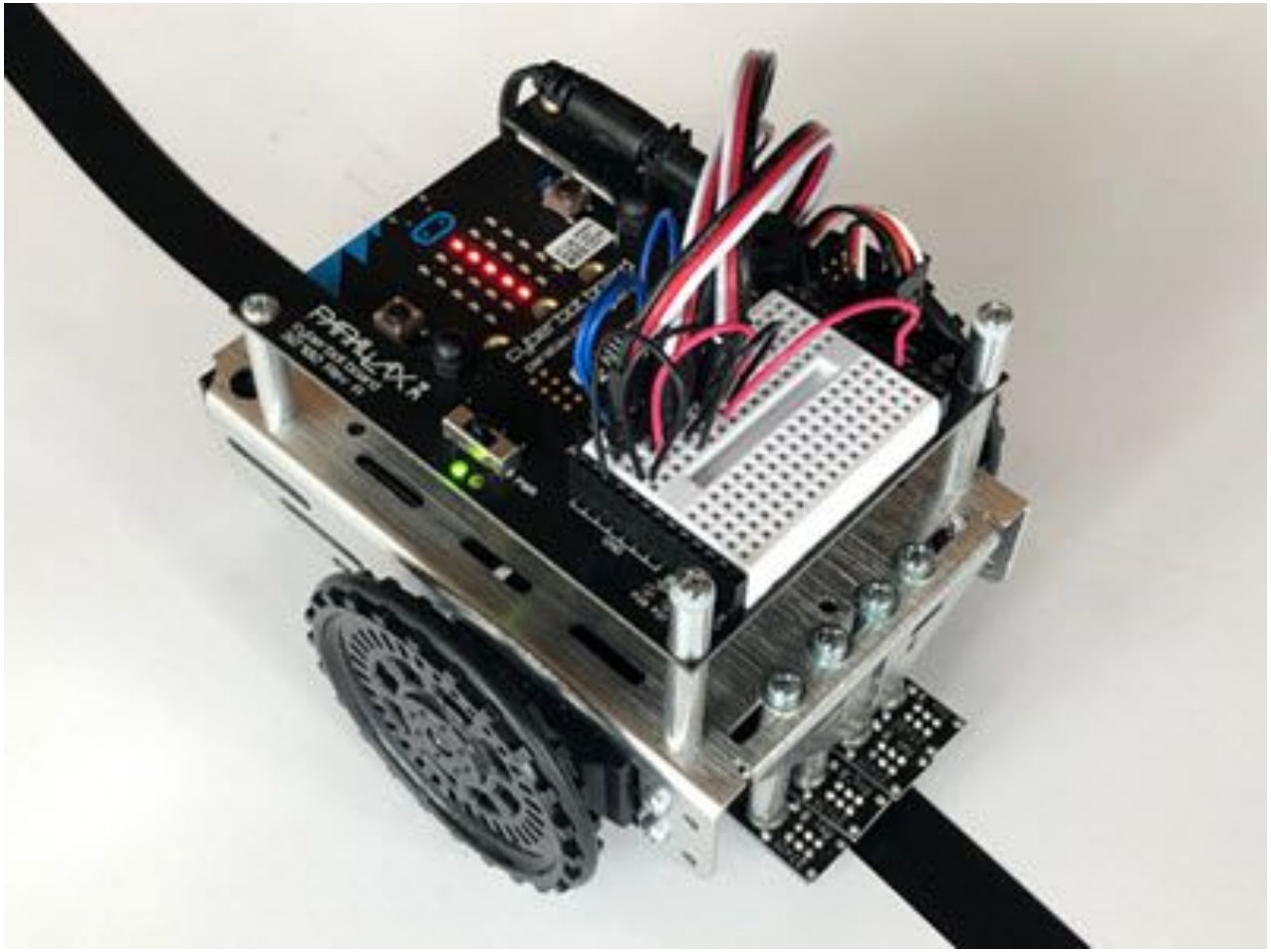
QTI Line Follower for cyber:bot

Control your cyber:bot with an Infrared TV Remote

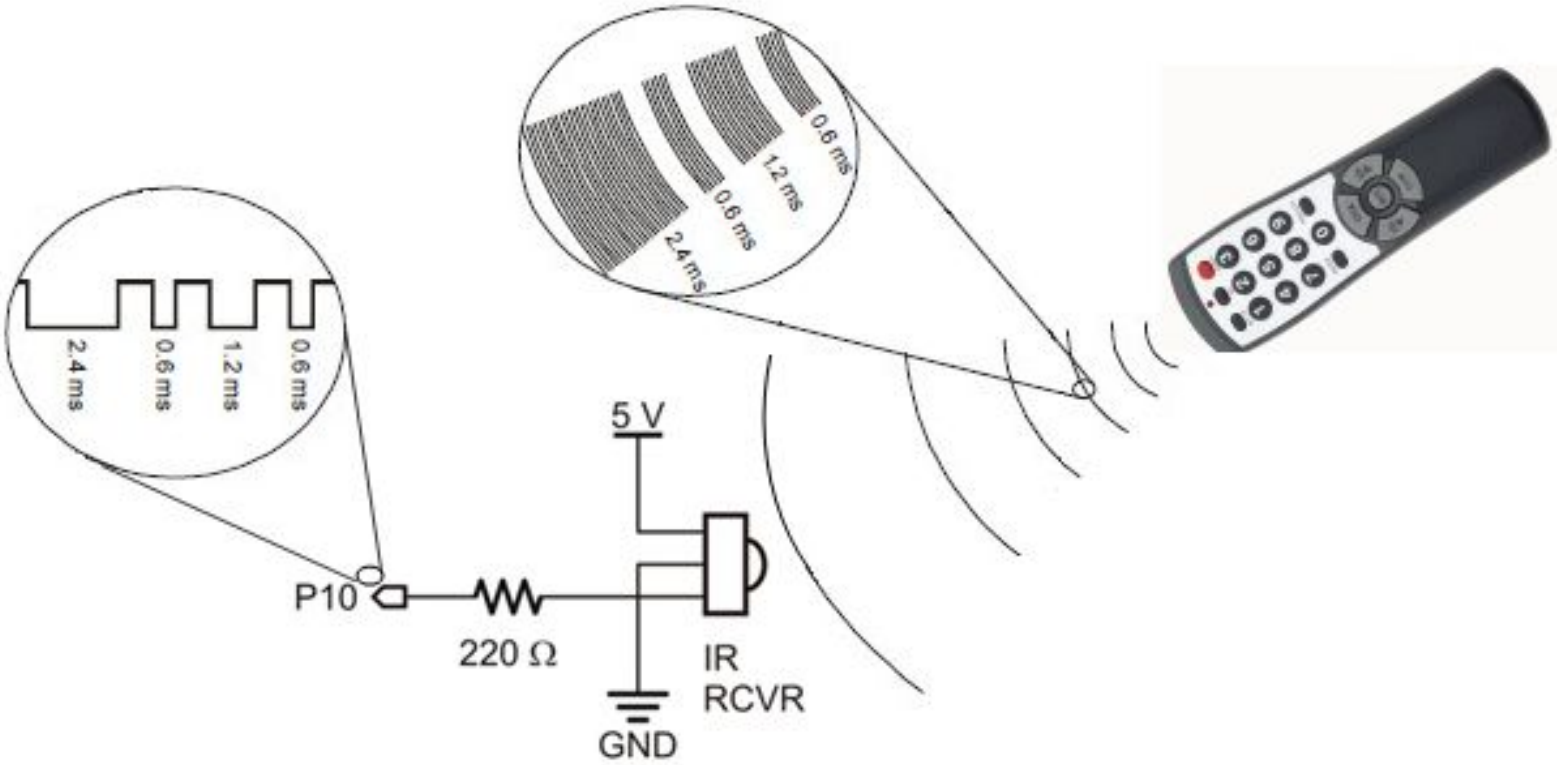


Projects

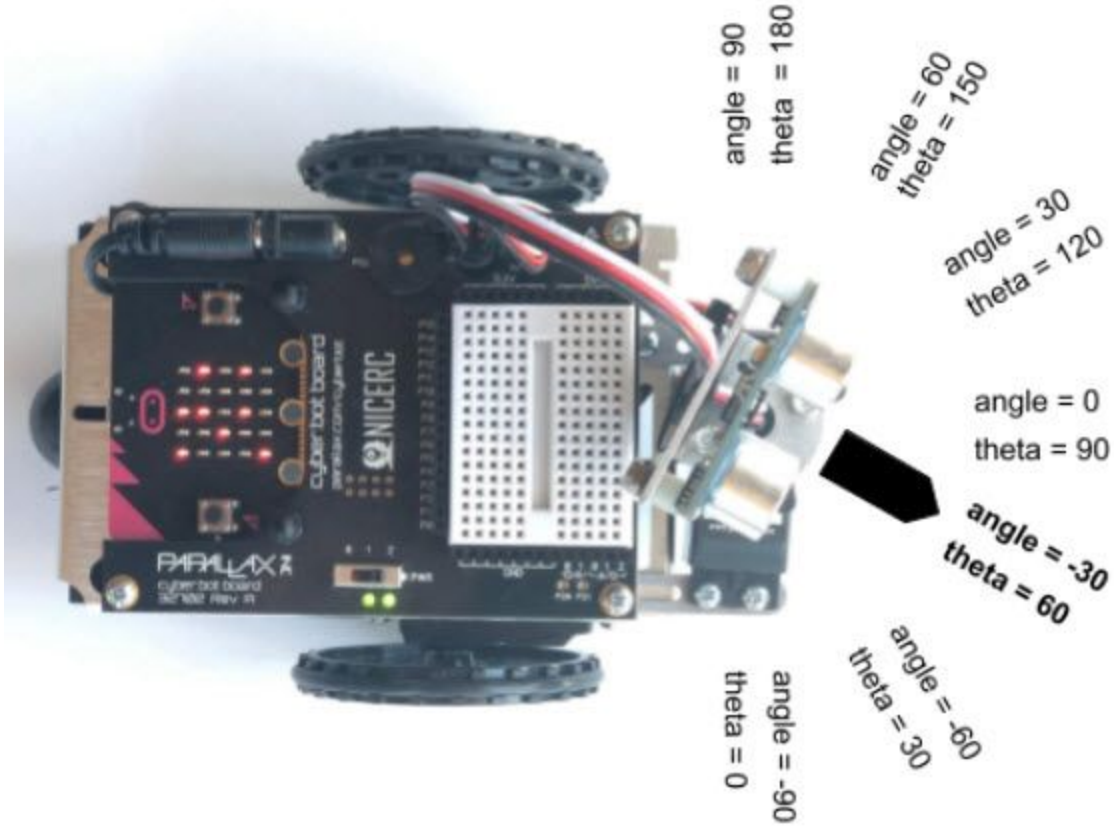
Line Follower

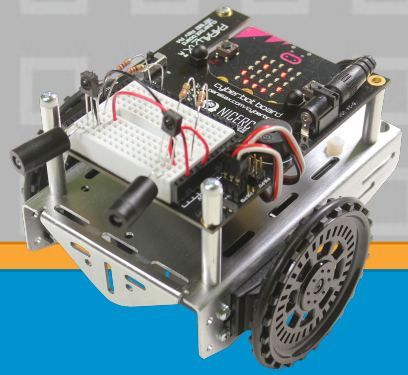


Infrared Remote Control



Roaming with the Ping))) Ultrasonic Sensor





Support

Purchasing cyber:bot

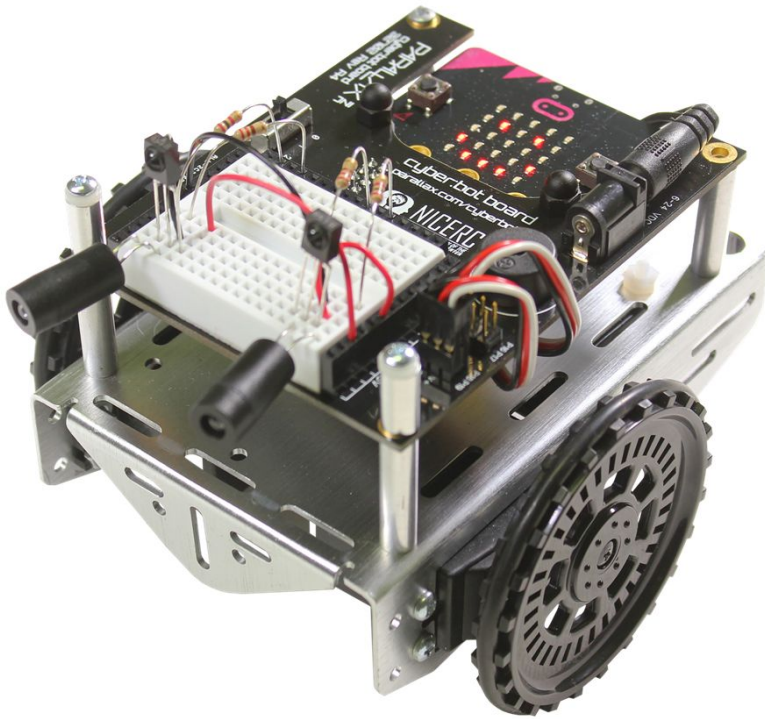
- Educator Hotline open 12 hrs/day (916) 701-8625
- E-mail learn@parallax.com
- Sales (916) 624-8333
- Forums http://forums.parallax.com/
- Facebook
  - Parallax https://www.facebook.com/ParallaxInc/
  - Micro:bithttps://www.facebook.com/groups/1756471244599979/

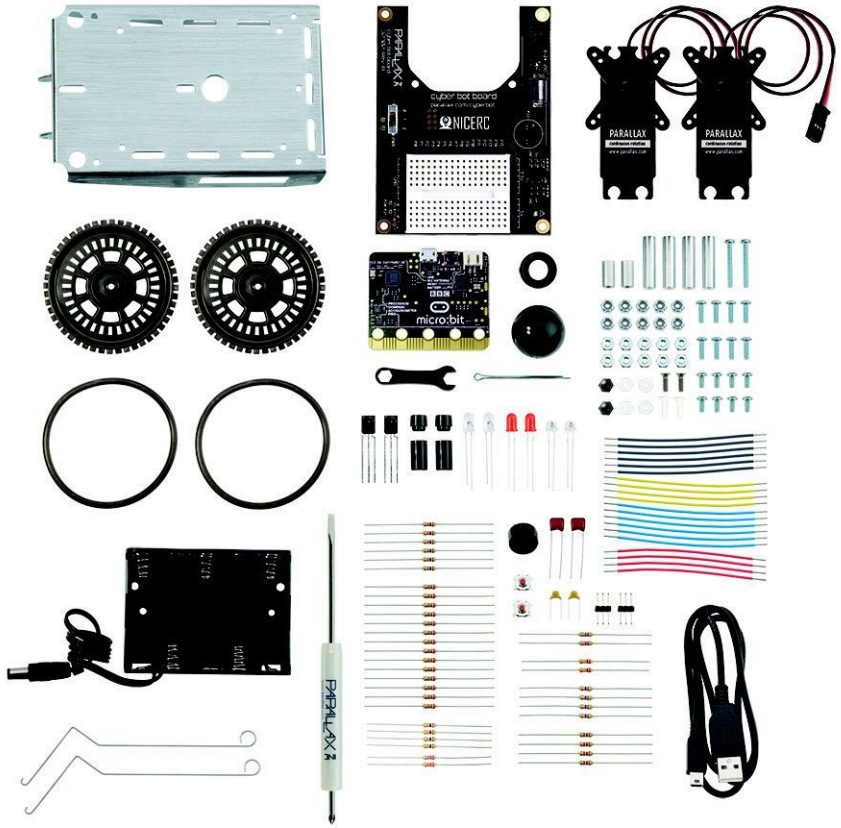
Cyber:bot with micro:bit #32700

$200 ea. (qty 1-9)

$190 ea. (qty 10-19)

$180 ea. (qty 20+)





Cyber:bot 12-Pack Plus #32700

$3,396.12 (regular $4,043.00)

- (12) cyber:bots
- (12) micro:bits
- (12) QTI Line Followers
- (12) Ping))) Ultrasonic sensors and servo mounting brackets
- (12) Infrared remote controls
- (5) battery chargers
- (60) NiMH batteries
- 2’x6’ class banner

Parallax Inc.
599 Menlo Drive, Ste 100
Rocklin, CA 95765

www.parallax.com
Learn.parallax.com
Main: (916) 624-8333
Educator Hotline: (916) 701-8625

National Integrated Cyber Education Research Center, NICERC
6300 East Texas Street
Bossier City, LA 71111

www.nicerc.org
Main: (318) 759-1600