Engineering Team,

We will be using the CuteBot and Micro:Bit to model Ampere's systems. This will be done largely based on the fact that the same type of technology will be implemented in the final product for ease of acquisition and assembly. We need you to develop the software for the motors that will allow the vehicle to move. The Ampere needs to be able to move forward, backward, left, and right. These capabilities are rudimentary, but allow more complex systems to guide the car when we develop them.

At your workstation, you should have the following:

- Project Envelope
- Watt Corp standard issue laptop (with charger)
- Micro:Bit
- CuteBot car
- 3x AAA Batteries
- 1x Micro-USB to USB-A cable

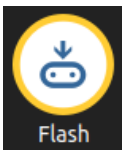
Please take a moment to familiarize yourself with the equipment around you. When you are ready, open your main.py and cutebotcar.py files in Mu – we will be adding on to them today.

First, select main.py. We will be adding to our main.py file to count our button presses to figure out how many times it has been pressed.

REMINDER – everything in grey is code you have already typed. You need only worry about the things in black or color.

```
from microbit import *
import cutebotcar as car

minLight = 5
display.show(Image.HAPPY)

a_count = 0
b_count = 0

while True:
    light = display.read_light_level()
    if light <= minLight:
        car.toggleHeadlights()
    elif (light > minLight) and car.headlightState:
        car.toggleHeadlights()

    if button_a.is_pressed():
        display.show(Image.SAD)

        sleep(1000)
        a_count = button_a.get_presses()
        if a_count == 2:
            display.scroll(str(a_count))

    elif button_b.is_pressed():
        display.show(Image.SILLY)
        car.toggleHeadlights()
        sleep(1000)

        b_count = button_b.get_presses()
        if b_count == 1:
            display.scroll(str(b_count))

    else:
        display.show(Image.HAPPY)
```


Click the save button! It is important that your edits to main.py get preserved so the car can operate as intended!




Next, put your Micro:Bit in your car and hook up your Micro:Bit to your PC. Then click the flash button! In the bottom left corner of Mu Editor, it should tell you if the flashing process failed or succeeded.

If you need help with this step because your computer isn't working, ask us : )

Now, push the reset button on the back of your Mirco:Bit. You should see a smiling face. When you push button A, your Micro:Bit should be sad. Quickly push it one more time. It should show you the amount of times you pressed the A button. If you push the B button once, it will also print the amount of times the button has been pressed to the screen. These are state changes, and we will use these state changes to dictate what our car is doing.

Open cutebotcar.py and make these additions.

```
from microbit import *

# --- BUILD I2C DRIVERS AND CONSTANTS ---
I2CAddr = 16

# HEADLIGHTS
leftLED = 4
rightLED = 8
headlightState = False

# MOTOR VALUES
leftMotor = 1
rightMotor = 2
spinForward = 2
spinBackward = 1
fullSpeed = 85
maxSpeed = 100
minSpeed = -100

def ledHelper(LED: int, headlightState: bool):
    [...]

def toggleHeadlights():
    [...]
```

These are our constants for the motors that we will be making. You can change the value of fullSpeed to the default value that you would like, but I chose to use 85.

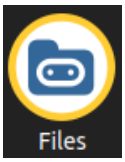
IMPORTANT: you will need to add the next code block (on the next page) be under your toggleHeadlights() function.

```
def speedHelper(motorPosition: int, speed: int):
    buffer = bytearray(4)
    buffer[0] = motorPosition

    # Test for extreme cases, then correct as needed
    if speed > maxSpeed:
        speed = maxSpeed
    elif speed < minSpeed:
        speed = minSpeed

    # Set buffer values to write
    if speed >= 0:
        buffer[1] = spinForward
    else:
        speed = speed * -1
        buffer[1] = spinBackward

    buffer[2] = speed
    buffer[3] = 0

    i2c.write(I2CAddr, buffer)

def setSpeed(leftSpeed: int, rightSpeed: int):
    # Set the speed for the left motor using I2C address and desired value
    speedHelper(leftMotor, leftSpeed)

    # Set the speed for the right motor using I2C address and desired value
    speedHelper(rightMotor, rightSpeed)

def goForward():
    # CAR GOES VROOM!
    setSpeed(fullSpeed, fullSpeed)

def turnLeft():
    # dnour' thgir em nips uoY
    setSpeed(-fullSpeed, fullSpeed)
```

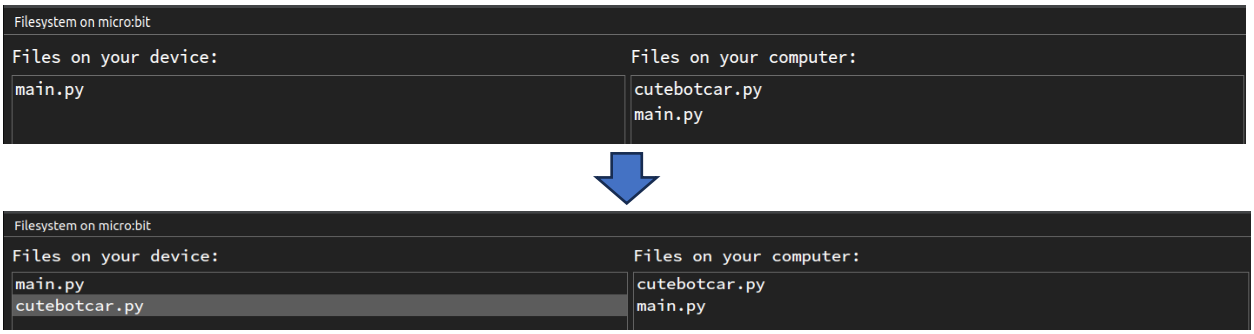
Here are the new pieces of code – they might look familiar to you. This is because ledHelper() that you wrote yesterday and speedHelper() are incredibly similar thanks to the I2C protocol! Just…instead of talking to LED's, we are talking to motors.

IMPORTANT: This code only gives you functions goForward() and turnLeft() – study these. You need to use them so you can make the functions goBackward(), turnRight(), and stop().


Click the save button! It is important that your edits to cutebotcar.py get preserved so the car can operate as intended!




We need to put our updated library file onto our car! Click the Files button! A window shoud pop up at the bottom of your screen. Find cutebotcar.py on the right and drag and drop it on the left. When the process is complete, main.py and cutebotcar.py should be on the left of the window. Click the files button again to close the window.



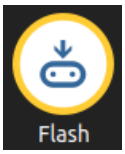

If you push the reset button on your Micro:Bit now, nothing will happen…know why? Well, we haven't used any of our new code in our main.py file. This is why started counting button presses! We need to be able to discern our car going left, right, forward, backward, and stop.

Now, open your main.py file again. We need to add the new functions we made and give our button counting states a few more options since our car can be in multiple states.

REMINDER: Everything in light grey you have already typed – so don't worry about that. Worry about everything that is in color or black on the next page.

```
from microbit import *
import cutebotcar as car

minLight = 5
display.show(Image.HAPPY)

a_count = 0
b_count = 0

while True:
    light = display.read_light_level()
    if light <= minLight:
        car.toggleHeadlights()
    elif (light > minLight) and car.headlightState:
        car.toggleHeadlights()

    if button_a.is_pressed():
        display.show(Image.SAD)

        sleep(1000)
        a_count = button_a.get_presses()
        if a_count == 2:
            car.goForward()
        else:
            car.stop()

    elif button_b.is_pressed():
        display.show(Image.SILLY)
        car.toggleHeadlights()
        sleep(1000)

        b_count = button_b.get_presses()
        if b_count == 1:
            car.turnLeft()
        elif b_count == 2:
            car.turnRight()
        else:
            car.goBackward()

    else:
        display.show(Image.HAPPY)
```


Click the save button! It is important that your edits to main.py get preserved so the car can operate as intended!




Next, put your Micro:Bit in your car and hook up your Micro:Bit to your PC. Then click the flash button! In the bottom left corner of Mu Editor, it should tell you if the flashing process failed or succeeded.

If you need help with this step because your computer isn't working, ask us : )

Congratulations! You now have a car that can locomote in four directions: forward, backward, left, and right! Push the A an B buttons respectively to change the direction that your car drives – don't let it get away from you though! Dr. Statham-Whittaker will be pleased to see your progress – and you're only two days into developing the Ampere! Unbelievable!

## BONUS ROUND

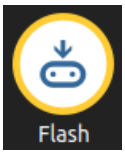
If you take notice, when the car starts moving forward or backward, it does so rather abruptly and jostles a little bit. Imagine us scaling the CuteBot up to the final Ampere product. If the automated system does the same thing, our passengers are going to file lawsuits against us for whiplash…so let's make the start and stop of our travel in the forward and backwards direction gentle by accelerating.

To do this, we will be utilizing the *for loop*, which performs a calculation *for* ever number *in the range* you specify. These instructions will use maxSpeed and minSpeed.

In main.py, add the following from the code block on the next page(everything in light grey you have already typed):

```
from microbit import *
import cutebotcar as car

minLight = 5
display.show(Image.HAPPY)

a_count = 0
b_count = 0

while True:
    light = display.read_light_level()
    if light <= minLight:
        car.toggleHeadlights()
    elif (light > minLight) and car.headlightState:
        car.toggleHeadlights()

    if button_a.is_pressed():
        display.show(Image.SAD)

        sleep(1000)
        a_count = button_a.get_presses()
        if a_count == 2:
            for speed in range(car.maxSpeed):
                car.setSpeed(speed, speed)
                sleep(100)
        else:
            car.stop()

    elif button_b.is_pressed():
        display.show(Image.SILLY)
        car.toggleHeadlights()
        sleep(1000)

        b_count = button_b.get_presses()
        if b_count == 1:
            car.turnLeft()
        elif b_count == 2:
            car.turnRight()
        else:
            for speed in range(car.maxSpeed):
                car.setSpeed(-speed, -speed)
                sleep(100)

    else:
        display.show(Image.HAPPY)
```


Click the save button! It is important that your edits to main.py get preserved so the car can operate as intended!




Next, put your Micro:Bit in your car and hook up your Micro:Bit to your PC. Then click the flash button! In the bottom left corner of Mu Editor, it should tell you if the flashing process failed or succeeded.

If you need help with this step because your computer isn't working, ask us : )

Whew – we dodged an expensive lawsuit there! Your CuteBot should now be modeling the basic functions of a mini Ampere. Now, report your code and findings to your managers!