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. Over the past couple of days, we have made a headlight system out of our LED's that detects ambient light and toggles as needed, we have learned how to write code for our motors to ensure we move forward, backward, left, and right. On top of those capabilities, we have created the ability to change car states with our button presses. Today, we will be combining all of the tools we have used to create ***AUTONOMOUS ELECTRIC CARS***. We will start by creating line sensing and following, which is what we will use to ensure our full-scale Ampere stays in the appropriate lanes. Once we polish that system, we will create ultrasonic sensing capabilities, which enables our cars to detect and avoid objects.

At your workstation, you should have the following:

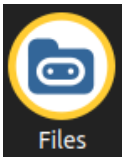
- Project Envelope
- Watt Corp standard issue laptop (with charger)
- Micro:Bit
- CuteBot car
- 3x AAA Batteries
- 1x Micro-USB to USB-A cable

Please take a moment to familiarize yourself with the equipment around you. Open cutebotcar.py. We will be adding line sensing to our library file so we have the ability to find the road and lanes below the car. Under your MOTOR VALUES block, add ULTRASONIC VALUES. REMINDER – everything in grey is code you have already typed. You need only worry about the things in black or color.

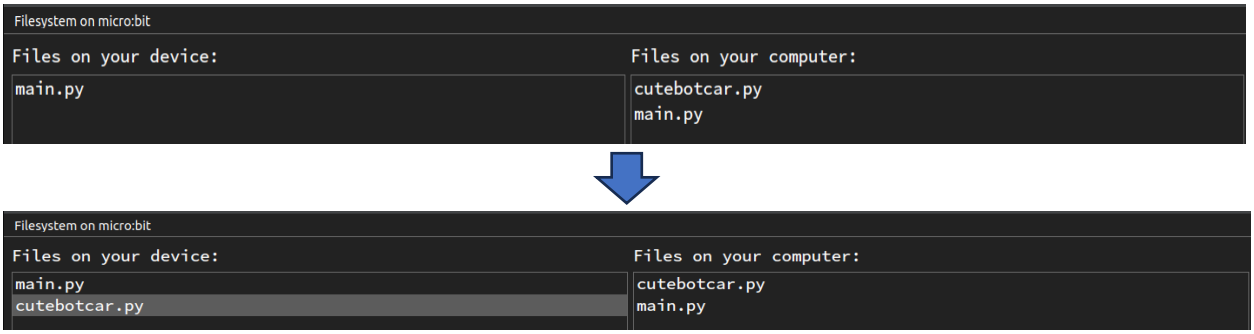
```
from microbit import *
from machine import time_pulse_us

[...]

# MOTOR VALUES
[...]

# ULTRASONIC VALUES
trigger = pin8
echo = pin12
trigger.write_digital(0)
echo.read_digital()
```

```
[ PLACE THIS CODE BLOCK BELOW EVERYTHING IN CUTEBOTCAR.PY]
def leftTrackSense():
    # If the left sensor sees the track, return TRUE!
    leftTrack = bool(pin13.read_digital())
    return not leftTrack

def rightTrackSense():
    # If the right sensor sees the track, return TRUE!
    rightTrack = bool(pin14.read_digital())
    return not rightTrack

def lineDetect():
    # This function detects the line beneath it and follows the line!
    if not leftTrackSense() and rightTrackSense():
        # Turn a little to the left to center the line
        setSpeed(40, 10)
    elif leftTrackSense() and not rightTrackSense():
        # Turn a little to the right and center the line
        setSpeed(10, 40)
    elif leftTrackSense() and rightTrackSense():
        # Line is in the middle of the sensors - onward!
        setSpeed(30, 30)
```

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.

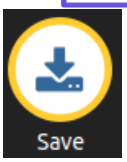


How does the code work? Well, leftTrackSense() and rightTrackSense() uses the IR sensor below the cutebot to detect a dark line. We can see if it is present or not by the little blue LED's on the front of the car. When they are illuminated, the car senses the road/lane beneath it. When they are off, the car can't find the road/lane and needs to adjust, which is why we use these values of the trackSense functions into the lineDetect() function.

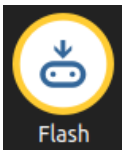
Cool, we have updated our library file, but we need to update our main.py file to use these new functions and give our car the ability to detect these roads/lanes.

NOTE: "[…]" just means all of your code here has not changed and to ignore this section

```
from microbit import *
import cutebotcar as car

[ ... ]

lineSensorDetect = False

while True:
    # Automatically detect the light and turn the lights on if needed!
    [ ... ]

    # If sensors have been enabled, engage them!
    if lineSensorDetect:
        car.lineDetect()

    # Check if our buttons are pressed.
    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)
        elif a_count == 3:
            car.lineDetect()
            lineSensorDetect = True
        else:
            car.stop()

    [ ... ]
```



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 3 times, your Micro:Bit should start automatically trying to detect a track. Ask us for one, and test your car out.

Feel free to fine-tune and change the values of your line following code. There are many ways you could modify it to be exactly what you think it should be – Watt Corp will review all of them and finalize the one that works the best.

Now, what if a child runs out in the middle of the street while our car is driving? We need to be able to detect objects in front of use and intelligently react. This is where we will use ***ULTRASONIC SENSING***. Essentially, the Ampere will detect objects similarly to how a dolphin does. We will make an ultrasonic noise, and measure how long it takes for that sound to return. If it does return, we can the use the time it took to receive it to calculate the distance of the object(s) in front of the Ampere.

Open cutebotcar.py and make these additions.

REMINDER: everything in grey is code you have already typed. You need only worry about the code in black and color.

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

```
def ultrasonicDistance(trigger, echo):
    # This function returns the range to find the distance between objects\\
    # Trigger burst
    trigger.write_digital(1)
    trigger.write_digital(0)

    # Try to catch the echo back, then convert to seconds
    microSeconds = time_pulse_us(echo, 1)
    echoTime = microSeconds / 1000000

    # Calculate distance in centimeters, then display
    distanceCM = (echoTime / 2) * 34300
    return int(distanceCM)

def ultrasonicSense():
    global trigger, echo
    distance = ultrasonicDistance(trigger, echo)

    # If the object is closer than 30 centimeters to the car
    if (distance <= 30):
        # Back up to assess the situation
        stop()
        sleep(200)
        goBackward(50, 50)
        sleep(500)

        # Check left
        turnLeft(0, 45)
        sleep(400)
        stop()
        leDistance = ultrasonicDistance(trigger, echo)
        turnLeft(0, -45)
        sleep(400)

        # Check right
        turnRight(45, 0)
        sleep(400)
        stop()
        riDistance = ultrasonicDistance(trigger, echo)
        turnRight(-45, 0)
        sleep(400)

        # Compare the distances: find which one is BIGGER
        if (leDistance >= riDistance):
            turnLeft(0, 45)
            sleep(400)
            goForward()
        else:
            turnRight(45, 0)
            sleep(400)
            goForward()
    else:
        goForward()
```

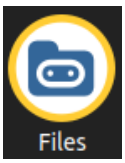
WAIT!!! Did you notice something different about our goForward, goBackward, turnLeft, and turnRight functions?

…they have…numbers in the parenthesis now…? Why?

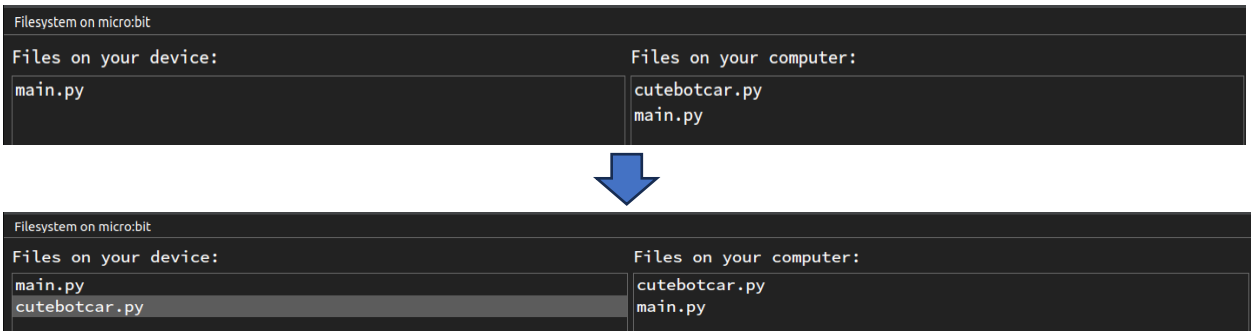
Well, when we what to automate the car, we need agreater sense of precision control, so we now need to make sure our goForward, goBackward, turnLeft, and turnRight functions can do that. Make the following changes.

```
def goForward(lefMotor: int = fullSpeed, rigMotor: int = fullSpeed):
    # CAR GOES VROOM!
    setSpeed(lefMotor, rigMotor)
def goBackward(lefMotor: int = fullSpeed, rigMotor: int = fullSpeed):
    # CAR GOES VROOM!
    setSpeed(-lefMotor, -rigMotor)

def turnLeft(lefMotor: int = -fullSpeed, rigMotor: int = fullSpeed):
    # dnour' thgir em nips uoY
    setSpeed(lefMotor, rigMotor)
def turnRight(lefMotor: int = fullSpeed, rigMotor: int = -fullSpeed):
    # You spin me right 'round
    setSpeed(lefMotor, rigMotor)
```

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



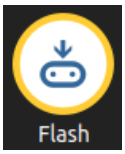
We need to put our updated library file onto our car! Click the Files button! A window shoud pop up at the bottom of your screen. Find cutebotcar.py on the right and drag and drop it on the left. When the process is complete, main.py and cutebotcar.py should be on the left of the window. Click the files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



These changes allow us to specify what speeds we want our motors at for greater control where it's needed. If we don't specify, they default to a value fullSpeed. Now, open main.py

```
from microbit import *
import cutebotcar as car

[ ... ]

lineSensorDetect = False
ultrasonicSensorDetect = False

while True:
    # Automatically detect the light and turn the lights on if needed!
    [ ... ]

    # If sensors have been enabled, engage them!
    if lineSensorDetect:
        car.lineDetect()
    elif ultrasonicSensorDetect:
        car.ultrasonicSense()

    # Check if our buttons are pressed.
    if button_a.is_pressed():
        [ ... ]

    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()
        elif b_count == 3:
            car.ultrasonicSense()
            ultrasonicSensorDetect = True
        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 :)

Now you should be able to reset your Micro:Bit, push the B button 3 times, and engage ultrasonic object detection! Dr. Statham-Whittaker is astounded by the progress you are making – keep up the good work!

BONUS ROUND

If you notice, ultrasonicSensorDetect() and lineDetect() operate separately...your bonus round, or should is say, ***CHALLENGE***, is to make both of these functions work together such that the Ampere can detect and object, deviate from the road/lane if possible, and after passing the object, merge back on to the road/lane.

There is no code that has been written for this application yet – *you will be the first* ***should you choose to accept this challenge.***