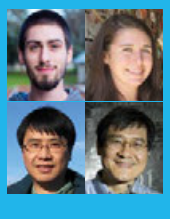
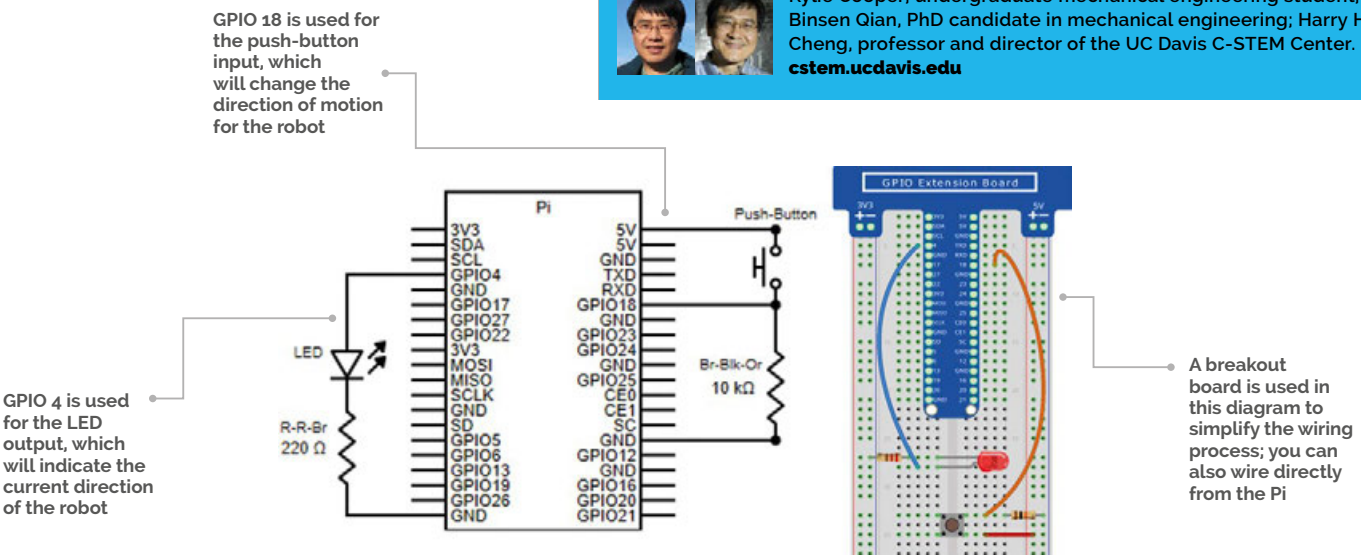


KYLE GOFF, KYLIE COOPER, BINSEN QIAN, HARRY H. CHENG

Kyle Goff, undergraduate computer engineering student; Kylie Cooper, undergraduate mechanical engineering student; Binsen Qian, PhD candidate in mechanical engineering; Harry H. Cheng, professor and director of the UC Davis C-STEM Center.
cstem.ucdavis.edu



CONTROLLING LEGO MINDSTORMS THROUGH GPIO

Construct a circuit to directly control a LEGO robot with the Raspberry Pi GPIO

You'll Need

- LEGO Mindstorms robot (NXT or EV3)
- Breadboard
- Wires for breadboard
- 1 × LED
- 1 × Push-button
- 1 × 220Ω resistor (Red-Red-Brown)
- 1 × 10kΩ resistor (Brown-Black-Orange)

LEGO Mindstorms is a great tool to gain experience in understanding robotics, but what if you wanted to make your own input sensor? In this guide, we will show how simple it is to construct a circuit to control a Mindstorms robot through GPIO in Raspberry Pi.

We will show every step from connecting the robot to writing the code. The result will be a program in Ch, a superset interpreter of C/C++, to control the direction of the robot with a push-button.

Software

To make use of C-STEM's programming tools, you should install the C-STEMbian operating system, which contains C-STEM Studio. This free, open-source operating system contains all the necessary tools for robotics and physical computing. Additionally, it is a superset of Raspbian, so all the familiar features will still be there. If you already have Raspbian installed, the C-STEM modules can be installed separately on top. All of this is available from the C-STEMbian section of the C-STEM website (**magpi.cc/2p3JUNP**). Step-by-step guides will assist you in setting up and accessing the Raspberry Pi if needed.

Connecting to the Mindstorms robot

Connecting to your Mindstorms robot is quite simple with the C-STEM software.

First, you will need to open C-STEM Studio and launch Ch Mindstorms Controller. Find the big 'C' at the top of the screen after logging in to your Raspberry Pi. Click the 'C', then navigate to 'Ch Mindstorms Controller' on the left side of the menu in C-STEM Studio. Click on Launch to open it.

Ch Mindstorms Controller can connect with both EV3 and NXT robots. Simply press the Scan Robot

Language
>CH

DOWNLOAD:
magpi.cc/2mHpaF4

TROUBLESHOOTING WITH GPIOVIEWER

Remember to use GPIOviewer to test your circuit before programming. You can test both the LED and the push button with the GUI.

button and add the robots that are found to the list on your robot manager. Follow the instructions on screen to pair the robots with your Raspberry Pi. Due to the limitations of Bluetooth, the Ch Mindstorms Controller can connect to a maximum of seven robots at a time. (Do make sure that the robots are turned on and have Bluetooth enabled!)

Once the robots have been scanned and added to the list, select the ones you would like to connect to and press Connect. Robots that you are connected to will have a green dot next to their names.

Building the simple circuit

The program in this tutorial requires a physical circuit to function. Our circuit will consist of a push-button input to control the direction of the robot's movement. An LED output will give a visual indication of the direction change when pressing the button.

Looking at the circuit, there are two sides: input and output. The input side, shown on the right, has a push button in series with a 10 kΩ resistor. The push-button is connected to 5 V for power. GPIO 18 is connected between them to read the button input.

The output side, on the left, has an LED in series with a 220 Ω resistor. GPIO 4 controls this light.

If you have one, use a breakout board to make the wiring process clearer. Otherwise, wire the pins directly from the Pi. Take a wire from GPIO 4 and connect it to an empty row of the breadboard. Then, attach the positive (longer) leg of an LED to this row. From the negative leg of the LED, attach a 220 Ω (Red-Red-Brown) resistor to ground.

For the push-button, insert it over the breadboard gutter. Wire 5 V to one lead, and wire a 10 kΩ (Brown-Black-Orange) resistor from ground to the adjacent leg. Finally, connect a wire from GPIO 18 to the row of the resistor and push-button leg. This will carry the input signal when the button is pressed.

Before programming, we can use GPIOviewer, a helpful feature of the C-STEMbian operating system. To use it, navigate again to the big 'C' at the top of the desktop window.

Once open, navigate to Ch Raspberry Pi and click Launch in the bottom right-hand corner. This will open up GPIOviewer, which allows total control of all the GPIO pins on the Raspberry Pi. In this view, you can change pin modes between input, output, and PWM (with a slider).

For this circuit, find GPIO 4 and set it to output.

Ensure the LED is set up and working properly by switching between high and low outputs. If the light turns on, you can move on to testing the input. Set GPIO 18 to input mode. Then, try pressing the button. If the input changes, the circuit is now ready for programming.

Coding in Ch

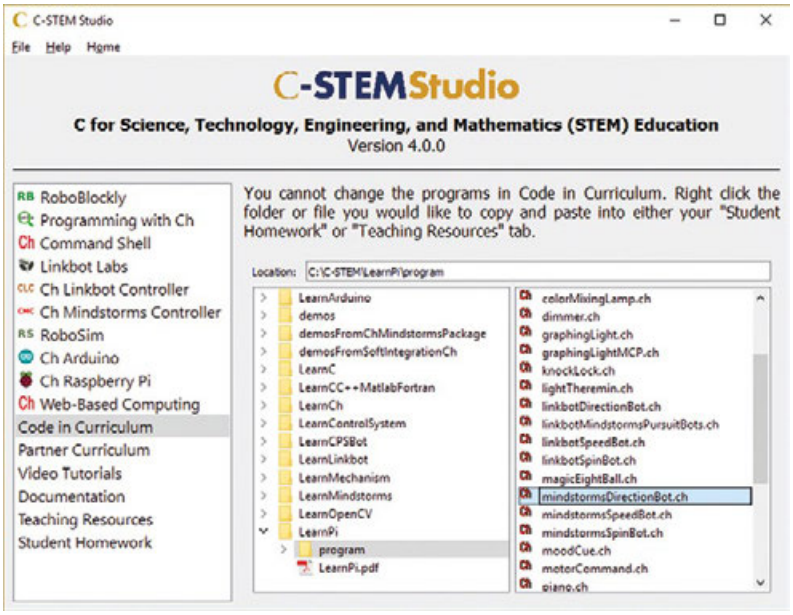
Programming in Ch starts by opening C-STEM Studio again on your Raspberry Pi. In v4.0, Navigate to Code in Curriculum > LearnPiprogram > mindstormsDirectionBot.ch. If you would like to make changes to the file, be sure to copy and paste it to another location before opening! To open the program with ChIDE, simply double-click it. The code for the project follows, which can be modified within the editing pane.

When running the code, be sure that the Mindstorms robot is still connected through CMC! Otherwise, the IDE will not recognise that

> When running the code, be sure that the Mindstorms robot is still connected through CMC!

the bot is connected and therefore will not run the code on it. The code should drive the robot forward or backward continuously at a constant speed. When the user gives input via a button press, the robot should switch directions by negating its speed. The LED will also change states on a button press by checking if the speed is positive or negative. Let's take a closer look at the Ch code to understand how this is done...

The first thing to notice are the two headers, **wiringPi.h** and **mindstorms.h**. We use the wiringPi header to take inputs and outputs more easily from



Open mindstormsDirectionBot.ch from the 'Code in Curriculum' section of C-STEM Studio

FOR HELP AND NEW IDEAS

Explore the free 'Learn Physical Computing with Raspberry Pi' and 'Learning Robot Programming with LEGO Mindstorms for the Absolute Beginner' textbooks in C-STEM Studio.

Check that the LED is working correctly by switching between high and low outputs on GPIOviewer

Check that the push-button is working correctly by making sure the input value changes after the button is pushed

the GPIO pins. It allows the code to resemble Arduino code. To use the Mindstorms platform, we include **mindstorms.h** which contains all robot functionality like moving and turning.

CMindstorms robot;

Now, the program needs to declare a robot. In Ch, **CMindstorms** is a class that can be instantiated like a variable. For this program, the Mindstorms robot is referred to as 'robot'; all functions related to it will be called using the **robot.function** format.

```
double radius = 1.1;
double speed = 5.0;
```

The first variable holds the radius of the wheels on the Mindstorms robot in units of inches. Remember that the radius of a circle, or wheel, is the distance from its centre to the edge. After that, a variable for speed is set to 5 inches (12.5cm) per second. Depending on the Mindstorms wheels attached, the radius may be different. By storing the value in a variable and passing it to functions, the code is easily adaptable to various sizes. If unsure, the radius of the physical wheel can be measured, especially if using a custom wheel.

```
int switchVal;
int directionPin = 18;
int ledPin = 4;
wiringPiSetupGpio();
pinMode(directionPin, INPUT);
pinMode(ledPin, OUTPUT);
```

Variables for the current switch value as well as the input/output pins are declared similarly to previous projects. Then, the pins are set up and initialised.

robot.driveForeverNB();

Before entering the **while** loop, the robot is set to continuously move using the **driveForeverNB()** function in the **CMindstormsI** class. It will drive forever in whatever direction it is currently facing. Forcing the robot to move constantly makes this code and physical circuit easier since the only input needed is the direction change. It is important not to use the **driveForever()** version without the 'NB' letters. For these functions, 'NB' stands for 'non-blocking', which allows the code to continue after the function has been called. Without the 'NB,' the code would stop at the function because it 'blocks' the program from continuing until it finishes.

```
while(1){
switchVal = digitalRead(directionPin);
delay(50);

if (switchVal == HIGH) {
speed = -speed;
robot.setSpeed(speed,radius);
robot.driveForeverNB();
}
```

The first section inside the infinite **while** loop checks the direction-changing pin. There is a **delay(50)**, meaning wait 50 milliseconds, to ensure a clean reading of the pin. Without this, it may switch directions multiple times on a single press. If the pin reads a value of 'HIGH' or '1', it will reverse the direction of movement. To accomplish this, the speed is set equal to its negative counterpart. For example, if the speed was 5 inches/second, this will change it to -5 inches/second. Therefore, the Mindstorms robot will move just as fast in either direction. Writing a new speed to the robot also requires the **setSpeed()** function in the **CMindstormsI** class. Notice that this function also requires the radius of the wheel because it uses this value to calculate how fast the wheel must spin to achieve the correct distance. Finally, one more **robot.driveForeverNB()** call is made to ensure the robot continues to move.

```
if (speed >= 0) {
digitalWrite(ledPin, HIGH);
}
else {
digitalWrite(ledPin, LOW);
}
}
```

To end the **while** loop, an **if** statement controls the state of the LED. When the robot's speed is greater than zero, it must be moving forward. In this case, the LED turns on. The LED turns off while the robot is moving backwards by checking if speed is less than zero.

If you want to take this project a step further, try connecting multiple robots and control them with the same circuit! Additionally, you can add LED traffic lights and make the robot move according to the lights. Or, come up with your own idea! Now you have the tools to make circuits that can interact with robots.