7

Python และ Arduino พร้อม Pyrmata

7.1 Python พร้อม Arduino

Arduino เป็นแพลตฟอร์มโอเพ่นซอร์สเพื่อสร้างสภาพแวดล้อมฮาร์ดแวร์และซอฟต์แวร์ Arduino มอบความเป็นไปได้ที่ไร้ขีดจำกัดสำหรับนักประดิษฐ์และผู้ที่ชื่นชอบอุปกรณ์อิเล็กทรอนิกส์

Raspberry Pi เป็นคอมพิวเตอร์แบบฟูลเอจที่สามารถทำงานได้เหมือนกับพีซีเดสก์ท็อป เป็นแพลตฟอร์มสำหรับการเข้ารหัสและออกแบบวงจรอิเล็กทรอนิกส์ ตั้งแต่การสร้างเว็บเซิร์ฟเวอร์ไปจนถึงคอนโซลเกมสำหรับการเล่นเกมย้อนยุค

Arduino ไม่เข้าใจ Python ดังนั้นจึงใช้โปรโตคอล Firmata และ Pyrmata เพื่อสื่อสารผ่าน Raspberry Pi โดยใช้ Python Pyrmata เป็นโปรโตคอลสำหรับ Raspberry Pi เพื่อเข้าถึง Arduino Firmata เป็นโปรโตคอลสำหรับ Arduino เพื่อเชื่อมต่อกับ Raspberry Pi กับ Python โปรแกรมจะถูกเขียนบน Raspberry Pi ใน Python เพื่อเข้าถึงเซ็นเซอร์ที่เชื่อมต่อกับ Arduino

ในการติดตั้ง Firmata เข้ากับ Arduino ให้เชื่อมต่อเข้ากับช่องเสียบ USB ของ Raspberry Pi เพื่อสื่อสารและเพิ่มพลังให้ Arduino ถัดไป ติดตั้ง Firmata Sketch ลงใน Arduino เพื่อเปิด Arduino IDE ค้นหาภาพสเก็ตช์ Firmata ในไฟล์→ตัวอย่าง→Firmata →มาตรฐาน Firmataและอัปโหลดไปยังบอร์ด Arduino เมื่อติดตั้ง Firmata แล้ว Arduino จะรอการสื่อสารจาก Raspberry Pi

ขั้นตอนต่อไปคือการติดตั้ง Pyrmata ลงใน Raspberry Pi สำหรับสิ่งนี้ เพียงเรียกใช้คำสั่งเทอร์มินัลต่อไปนี้บน Raspberry Pi:

```
$ sudo apt-get ติดตั้ง git
$ sudo git โคลนhttps://github.com/tino/pyFirmata.git $
cdpyFirmata
$ sudo python setup.py ติดตั้ง'
```

7.2 การควบคุม Arduino ด้วย Python

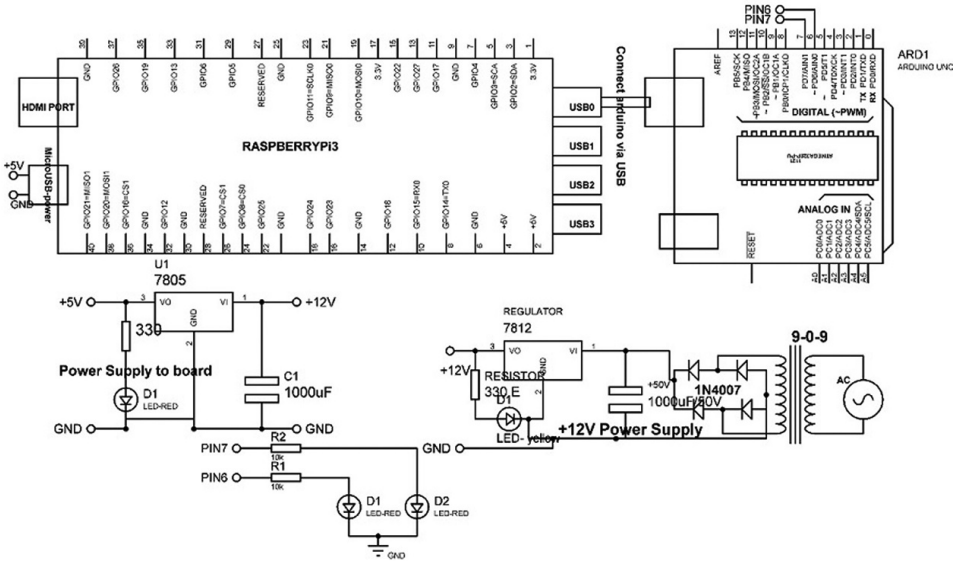
ข้วต่อ "USB มาตรฐาน A" ใช้สำหรับเชือมต่อ Arduino กับ Raspberry Pi ตรวจสอบทีอยู่ USB ของ Arduino โดยเรียกใช้ "ลส
- lrt /dev/tty*”ใน Raspberry Pi ของฉันมีรายการเป็น /dev/ttyUSB0 (จำค่านีไว้ใช้ภายหลัง)

นำเข้าคลาส Arduino และ util จากโมดูล Pyrmata เพือควบคุม Arduino จากสคริปต์ Python บน Raspberry Pi หลังจากนี ให้สร้างวัตถุทีพบในขันตอนก่อนหน้าโดยใช้ทีอยู่ USB

```
> > >จาก pyrmata นำเข้า Arduino util
> > >บอร์ด = Arduino('/dev/ttyUSB0')
```

7.3 เล่นกับ LED

วัตถุประสงค์ของโครงการนีคือการควบคุมเอาต์พุตดิจิตอล Arduino ผ่าน Raspberry Pi ด้วย Python ในการสร้างโปรเจ็กต์นี ให้เชือมต่อ LED กับพินดิจิทัลของ Arduino และเขียนโปรแกรม Python สันๆ เพือให้กะพริบรูปที่7.1 แสดงแผนภาพวงจรสำหรับการเชือมต่อของ LED ระบบประกอบด้วย Raspberry Pi3, Arduino Uno, แหล่งจ่ายไฟ และไฟ LED สองดวงทีเชือมต่อกับ Pin6 และ Pin7 ของ Arduino โปรแกรมถูกเขียนขึนเพือให้ไฟ LED กะพริบหลังจากหน่วงเวลาไประยะหนึง



รูปที 7.1
แผนภาพวงจรสำหรับเชือมต่อ LED

7.3.1 สูตร

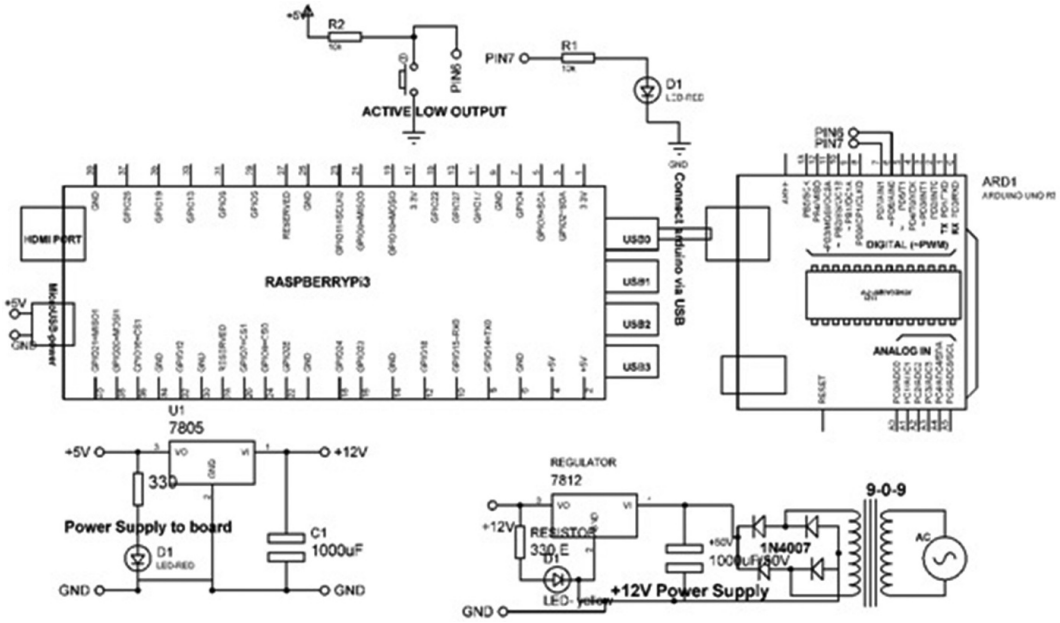
```
นำเข้า pyrmata # นำเข้า lib ของ pyrmata นำเข้าเวลา
ทีรอ # นำเข้า lib ของ pyrmata
board = pyrmata.Arduino('/dev/ttyUSB0')# dene COM port ของ Arduino
red_pin = board.get_pin('d:7:o')# กำหนดพินดิจิตอล 7 เป็นเอาต์พุต green_pin
= board.get_pin('d: 6:o')# กำหนดพินดิจิตอล 6 เป็นเอาต์พุต

ในขณะทีจริง:       #อินไนท์ลูป
        red_pin.write(1)# เขียน '1' บนพิน 7
        green_pin.write(1)# เขียน '1' บนพิน 6
        wait.sleep(0.5)# ล่าช้า 0.5 วินาที
        red_pin.write(0)#write '0' บนพิน 7
        green_pin.write(0)# เขียน '0' บนพิน 6
        wait.sleep(0.5)# ล่าช้า 0.5 วินาที
```

7.4 การอ่านอินพุตดิจิตอล Arduino ด้วย Pyfirmata

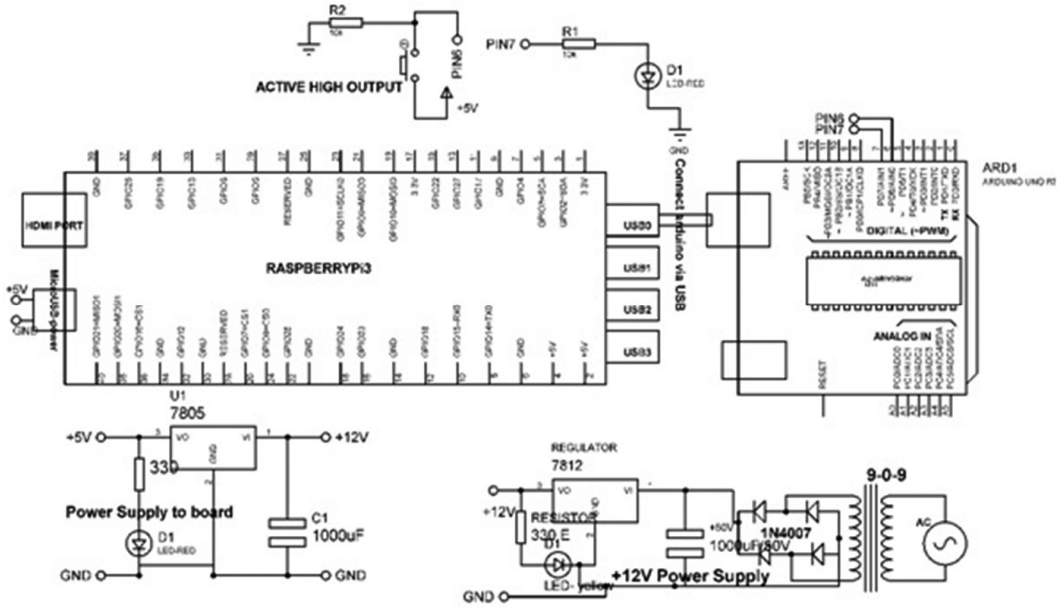
วัตถุประสงค์คือเพื่ออ่านพินดิจิตอลของ Arduino บน Raspberry Pi โดย Python Pyrmata ใช้สำหรับอ่านอินพุตดิจิตอลบน Arduino ส่วนประกอบที่จำเป็นสำหรับสูตรคือ Arduino Uno ตัวต้านทาน 1 kΩ และสวิตช์หรือปุ่มแบบกด (เป็นเซ็นเซอร์ดิจิทัล) สวิตช์สามารถเชื่อมต่อได้สองแบบ: ดึงลงและดึงขึ้น เอาต์พุตของพินดิจิทัลของ Arduino มักจะเป็น "LOW" และเซ็นเซอร์ดิจิทัลมีให้เลือกสองแบบสำหรับเอาต์พุต: แอ็คทีฟ "LOW" และแอ็คทีฟ "สูง" การจัดเรียงแบบดึงลงจะใช้โดยปกติพินดิจิทัลจะ "ต่ำ" และเมืออ่านเซ็นเซอร์จะได้รับ "สูง" ใช้สำหรับเซ็นเซอร์ที่มีเอาต์พุตเป็น "HIGH" ที่ใช้งานอยู่เมือเกิดเหตุการณ์ มิฉะน้นเอาต์พุตจะเป็น "LOW" การจัดเรียงแบบดึงขึ้นมีไว้สำหรับเซ็นเซอร์ที่มีเอาต์พุตปกติเป็น "สูง" ที่ใช้งานอยู่ และเมือเกิดเหตุการณ์จะได้รับ "ต่ำ"รูปที่ 7.2แสดงแผนภาพวงจรสำหรับดึงลงและรูปที่ 7.3แสดงแผนภาพวงจรสำหรับดึงขึ้น

ตามทีกล่าวไว้ในมาตรา IIของหนังสือเล่มนี้ โปรโตคอล Pyrmata ใช้เพืออ่านพินอินพุตของ Arduino โดย Raspberry Pi มันใช้แนวคิดของตัววนซ้ำเพือตรวจสอบพิน Arduino ตัววนซ้ำจัดการการอ่านสวิตช์โดยใช้คำสั่งต่อไปนี้:

```
มัน = pyrmata.util Iterator(บอร์ด)
it.start()
```



รูป 7.2
การจัดเรียงแบบดึงลงสำหรับปุ่มอ่าน



รูป 7.3
การจัดเรียงแบบดึงขึ้นสำหรับปุ่มอ่าน

หลังจากนี้เปิดใช้งานพินโดยใช้คำสั่งต่อไปนี้

```
switch_pin.enable_reporting()
```

ไม่สามารถหยุดฟังก์ชันตัววนซ้ำได้ ดังนั้นเมื่อกด Ctrl+Z เพื่อออกจากหน้าต่าง ฟังก์ชันนั้นจะไม่มีอยู่

หากต้องการหยุดฟังก์ชันนี้ ให้ยกเลิกการเชื่อมต่อ Arduino จาก Raspberry Pi หรือเปิดหน้าต่างเทอร์มินัลอื่นแล้วใช้คำสั่ง kill:

```
$ sudokillall python
```

7.4.1 สูตรการอ่านการจัดวางแบบดึงลง

```
นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาที่รอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # dene COM port ของ Arduino
button_pin = board.get_pin('d:6:i') # dene pin 6 เป็นอินพุต led_pin =
board.get_pin('d:7 :o') # dene pin7 เป็นเอาต์พุต
มัน = pyrmata.util.Iterator(บอร์ด) # ใช้ตัววนซ้ำ
it.start() # เริ่มตัววนซ้ำ
button_pin.enable_reporting() # เปิดใช้งานอินพุตใน
ขณะที่ True: # innite loop
        switch_state = switch_pin.read () # อ่านอินพุตจากพิน 6
        ifswitch_state == เท็จ: # ตรวจสอบเงื่อนไข
                print('Button Pressed') # print string บน Pi terminal
                led_pin.write(1) # write '1' on pin 7
                wait.sleep(0.2) # ล่าช้า 0.2 วินาที
อื่น
                พิมพ์ ('ไม่ได้กดปุ่ม') # พิมพ์สตริงบนเทอร์มินัล Pi
                led_pin.write (0) # เขียน '0' บนพิน 7
                wait.sleep(0.2) # ล่าช้า 0.2 วินาที
```

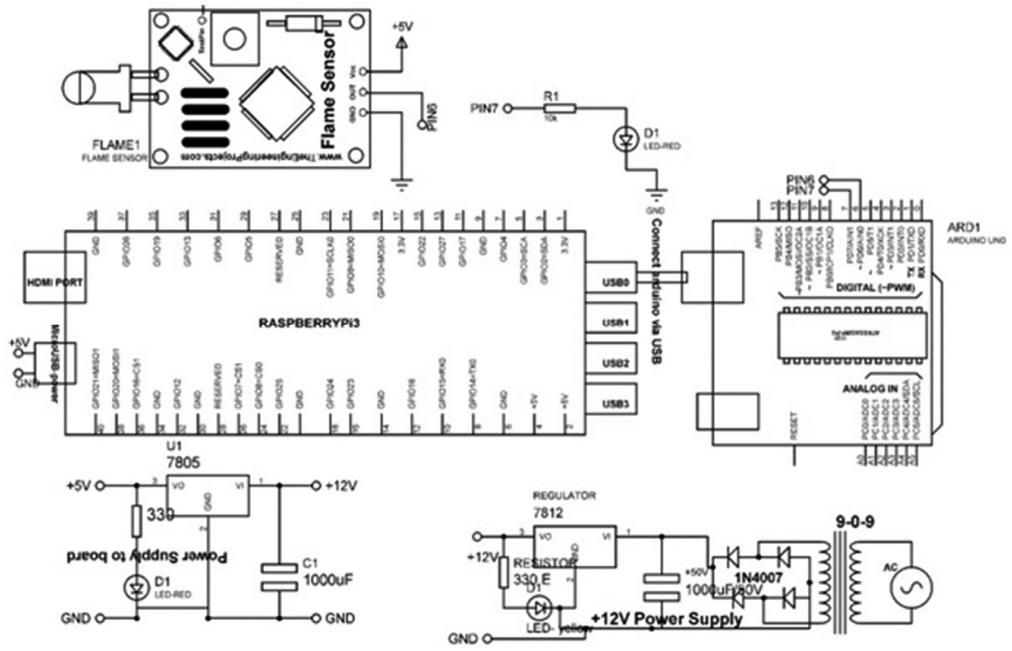
7.4.2 สูตรการอ่านการจัดวางแบบดึงขึ้น

```
นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาที่รอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # dene COMport ของ Arduino
button_pin = board.get_pin('d:6:i') # กำหนดพิน 6 เป็นอินพุตดิจิตอล led_pin =
board.get_pin('d:7: o') # กำหนดพิน 7 เป็นเอาต์พุตดิจิตอล
มัน = pyrmata.util.Iterator(บอร์ด) # ใช้ตัววนซ้ำ
it.start() # เริ่มตัววนซ้ำ
button_pin.enable_reporting () # เปิดใช้งานพินใน
ขณะที่ True: # innite loop
        switch_state = switch_pin.read() # อ่านพินดิจิตอล
        ifswitch_state == True: # ตรวจสอบเงื่อนไข
                print('Button Pressed') # print string บน Pi terminal
```

```
                led_pin.write (1) # ทำพิน 7 ถึง '1'
                wait.sleep (0.2) # ล่าช้า 0.2 วินาที
    อืน
                พิมพ์ ('ไม่ได้กดปุ่ม') # พิมพ์สตริงบนเทอร์มินัล Pi
                led_pin.write (0) # ทำพิน 7 ถึง '1'
                wait.sleep(0.2) # ล่าช้า 0.2 วินาที
```

7.5 การอ่านเซ็นเซอร์เปลวไฟด้วย Pyfirmata

วัตถุประสงค์คือเพื่ออ่านเซ็นเซอร์ ame เป็นอินพุตด้วย Python บน Raspberry Pi เซ็นเซอร์ ame สามารถตรวจจับแสงอินฟราเรดที่มีความยาวคลื่นตั้งแต่ 700 ถึง 1,000 นาโนเมตร โพรบ ame อินฟราเรดไกลจะแปลงแสงที่ตรวจจับได้ในรูปของแสงอินฟราเรดให้เป็นกระแส มีแรงดันไฟทำงาน 3.3 ถึง 5.2 V DC พร้อมเอาต์พุตดิจิตอลเพื่อบ่งชี้ว่ามีสัญญาณอยู่ ตัวเปรียบเทียบออนบอร์ด LM393 ใช้สำหรับการตรวจจับสภาพ เชื่อมต่อส่วนประกอบตามที่แสดงในรูปที่7.4และตรวจสอบการทำงานโดยอัพโหลดสูตรที่อธิบายไว้ในมาตรา 7.5.1.



รูป 7.4
แผนภาพวงจรสำหรับการเชื่อมต่อเซ็นเซอร์ ame

7.5.1 โปรแกรมสำหรับการอ่านเซ็นเซอร์เปลวไฟ "ต่ำ" ทีใช้งานอยู่

```
นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาทีรอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # dene COMport ofArduino
ame_pin = board.get_pin('d:6:i') # กำหนดพิน 6 เป็นอินพุตดิจิตอล
indicator_pin = board.get_pin('d:7:o ') # กำหนดพิน 7 เป็นเอาต์พุตดิจิตอล =
pyrmata.util.Iterator (บอร์ด) # ใช้ตัววนซ้ำ
it.start() # start iterator
ame_pin.enable_reporting() # เปิดใช้งานอินพุตใน
ขณะที True: # innite loop
        ame_state = ame_pin.read () # อ่านอินพุตดิจิตอล
        ifame_state == เท็จ: # ตรวจสอบเงือนไข
                พิมพ์ ('ไม่มีอุปสรรค') # พิมพ์สตริงบน Pi Terminal
                indicator_pin.write (1) # เขียน '1'on pin7 wait.sleep
                (0.2) # สลีปเป็นเวลา 0.2 วินาที
    อืน:
                พิมพ์ ("พบสิงกีดขวาง") # พิมพ์สตริงบน Pi Terminal
                indicator_pin.write (0) # เขียน '0' บน pin7
                wait.sleep(0.2) # sleep สำหรับ 0.2 วินาที
```

7.6 การอ่านอินพุตแบบอะนาล็อกด้วย Pyfirmata

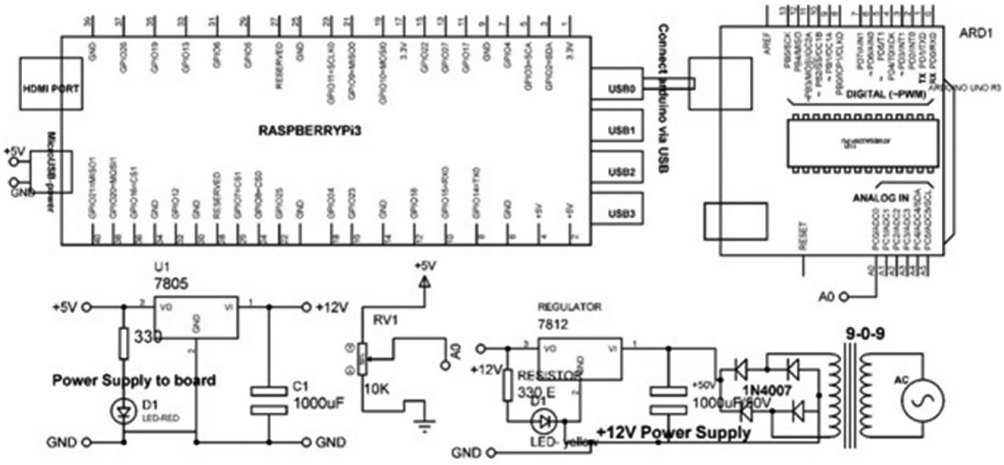
โพเทนชิออมิเตอร์ใช้เพือสาธิตการทำงานของเซ็นเซอร์แอนะล็อกด้วย Pyrmata เชือมต่อกับขา A0 ของ Arduino (รูปที่7.5). เมือพินได้รับการเชือมต่อเป็นพินอินพุตแบบอะนาล็อกในโปรแกรม จะเริมส่งค่าอินพุตไปยังพอร์ตอนุกรม หากไม่สามารถจัดการข้อมูลได้อย่างเหมาะสม ข้อมูลจะเริมบัฟเฟอร์ทีพอร์ตอนุกรมและโอเวอร์โอเวอร์อย่างรวดเร็ว สถานการณ์นีสามารถจัดการกับโปรแกรมได้

ไลบรารี Pyrmata มีวิธีการรายงานและตัววนซ้ำเพือเอาชนะสถานการณ์นี ดิ enable_reporting()เมธอดใช้เพือตังค่าพินอินพุตเพือเริมการรายงาน วิธีนีใช้ก่อนดำเนินการอ่านบนพิน:

```
board.analog[3].enable_reporting()
```

เมือดำเนินการอ่านเสร็จแล้ว หมุดจะถูกตังค่าให้ปิดใช้งานการรายงาน:

```
board.analog[3].disable_reporting()
```



รูปที่7.5
แผนภาพวงจรสำหรับการเชื่อมต่อโพเทนชิออมิเตอร์

หากต้องการอ่านพินอนาล็อกiteratorthreadใช้ในลูปหลัก
คลาสนี้ถูกปฏิเสธในโมดูล util ของแพ็คเกจ Pyrmata และนำเข้าก่อนที่จะนำไปใช้ในโค้ด:

จาก pyrmata นำเข้า Arduino util
# การตั้งค่าพอร์ตบอร์ด
Arduino = 'COM3'
บอร์ด = Arduino (พอร์ต)
นอน(5)
it = util.Iterator(board) # Start Iterator เพื่อหลีกเลี่ยงอนุกรม overow
it.start()
board.analog[3].enable_reporting()

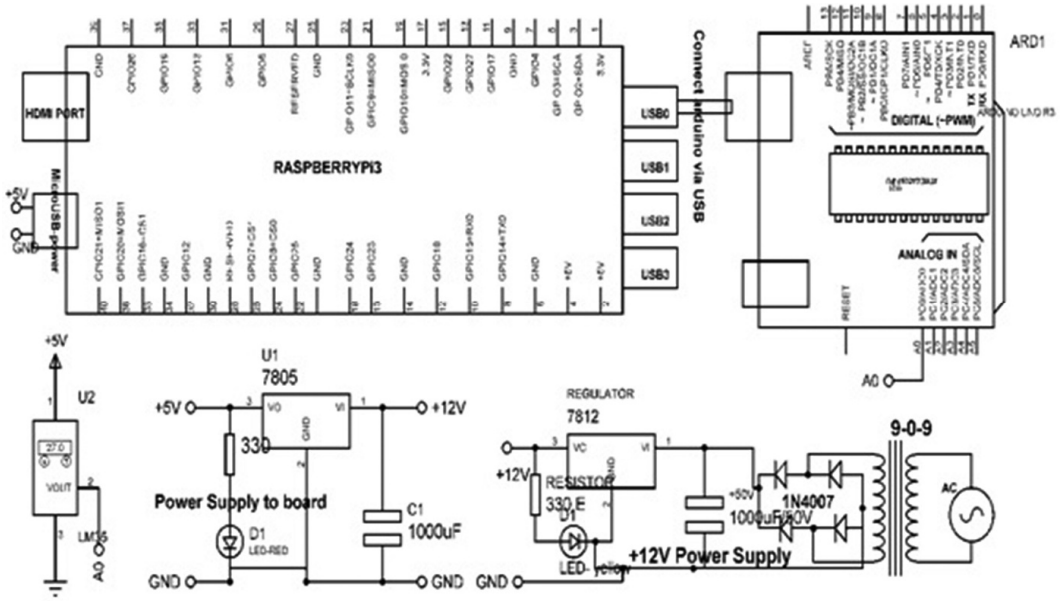
7.6.1 สูตร

นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาทีรอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # ปฏิเสธพอร์ต COM ของ Arduino
POT_pin = board.get_pin('a:0:i') # กำหนดพิน A0 เป็นอินพุต =
pyrmata.util.Iterator (บอร์ด) # ใช้ iterator
it.start() # start iterator
POT_pin.enable_reporting() # เปิดใช้งานพินใน
ขณะที่ True: # innite loop
POT_reading = POT_pin.read () # อ่านพินอะนาล็อก

```
if POT_reading != None: # ตรวจสอบเงื่อนไข
                POT_voltage = POT_reading * 5.0 # แปลงระดับเป็น
                    แรงดันไฟฟ้า
                พิมพ์("POT_reading=%f\t POT_voltage=%f"% (POT_
                    การอ่าน POT_voltage))
                # printvalues   บน Pi terminal
                wait.sleep(1) # sleep for 1 sec
อื่น:
            print("ไม่มีการอ่าน") # พิมพ์สตริงบน Pi terminal
            wait.sleep(1)# sleep for 1 sec
```

7.7 การอ่านเซ็นเซอร์อุณหภูมิด้วย Pyfirmata

เซ็นเซอร์อุณหภูมิซีรีส์ LM35 มีแรงดันเอาต์พุตตามสัดส่วนเชิงเส้นกับอุณหภูมิเซนติเกรด อุปกรณ์ LM35 ไม่ต้องการการสอบเทียบหรือการตัดแต่งเพื่อให้ความแม่นยำของ±¼°C ที่อุณหภูมิห้องและมีช่วงการตรวจจับที่ −55°C ถึง 150°C. อุปกรณ์ LM35 ดึงกระแส 60 μA จากแหล่งจ่าย อุปกรณ์ซีรีส์ LM35 มีจำหน่ายในแพ็คเกจทรานซิสเตอร์ TO แบบสุญญากาศ ในขณะที่ LM35C, LM35CA และ LM35D มีอยู่ในแพ็คเกจทรานซิสเตอร์ TO-92 แบบพลาสติกรูปที่7.6แสดงแผนภาพวงจรของการเชื่อมต่อ LM35 เอาต์พุตของ LM35 เชื่อมต่อกับพิน A0 ของ Arduino



รูป 7.6
แผนภาพวงจรของการเชื่อมต่อ LM35

7.7.1 สูตร

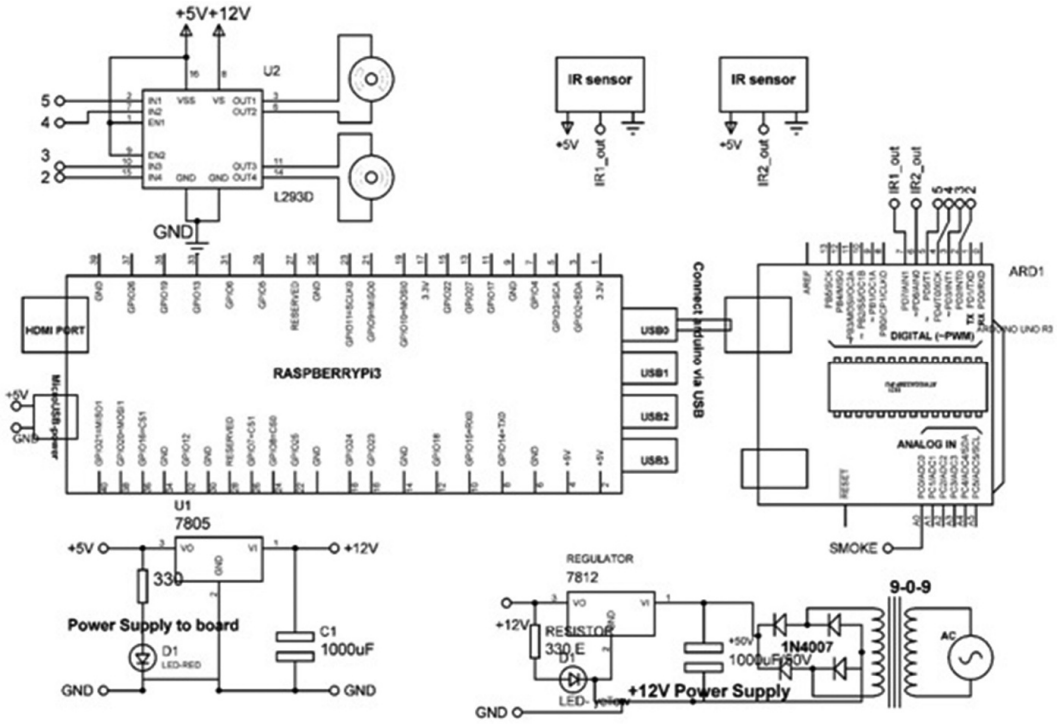
```
นำเข้า pyrmata # นำเข้าไลบรารีของเวลานำเข้า
pyrmata เนืองจากรอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # ปฏิเสธพอร์ต COM ของ
    Arduino
POT_pin = board.get_pin('a:0:i') # กำหนดพิน A0 เป็นอินพุต =
pyrmata.util.Iterator (บอร์ด) # ใช้ iterator
it.start() # start iterator
POT_pin.enable_reporting() # เปิดใช้งานพินใน
ขณะที่ True:
การอ่าน = switch_pin.read() # อ่านอินพุตแบบอะนา
ล็อกหากอ่าน != ไม่มี: # ตรวจสอบเงือนไข
                แรงดันไฟฟ้า = การอ่าน * 5.0 # แปลงระดับเป็นแรงดันไฟฟ้า
                อุณหภูมิ = (แรงดันไฟฟ้า*1000)/10 # แปลงแรงดันไฟฟ้าเป็น
                    อุณหภูมิ
                print('กำลังอ่าน=%f\t          แรงดันไฟฟ้า=%f\tอุณหภูมิ=%f'%
                    (การอ่าน แรงดัน อุณหภูมิ))
                # พิมพ์ค่าบน Pi Terminal
                wait.sleep(1) # sleep for 1 Sec
อืน:
                print("No readingObtained") # print string บน Pi Terminal
                wait.sleep(1)# sleep for 1 Sec
```

7.8 หุ่นยนต์เดินตามเส้นด้วย Pyfirmata

หุ่นยนต์ติดตามเส้นตามภาพบน oor หรือเพดาน โดยปกติ เส้นสายตาจะเป็นสีดำบนพื้นผิวสีขาว แม้ว่าเส้นสีขาวบนพื้นผิวสีดำก็เป็นไปได้เช่นกัน หุ่นยนต์ตัวตามสายการผลิตใช้ในอุตสาหกรรมการผลิตสำหรับกระบวนการอัตโนมัติ เป็นหนึงในหุ่นยนต์พื้นฐานทีสุดสำหรับผู้เริ่มต้น เพือให้เข้าใจการออกแบบหุ่นยนต์ด้วย Raspberry Pi และ Arduino Uno ระบบจึงประกอบด้วยตัวขับมอเตอร์ L293D มอเตอร์ DC สองตัว ล้ออิสระ (เพือเชือมต่อทีด้านหน้าของหุ่นยนต์) เซ็นเซอร์ IR สองตัว และกำลังไฟฟ้า จัดหา.

การเชือมต่อ:

- ต่อพิน (IN1, IN2, IN3, IN4) ของ L293D เข้ากับพิน (5, 4, 3, 2) ของ Arduino Uno ตามลำดับ
- เชือมต่อมอเตอร์กระแสตรง (M1) ระหว่างพิน (OUT1 และ OUT2) ของ L293D



รูปที่7.7
แผนภาพวงจรสำหรับหุ่นยนต์ตามเส้น

- เชื่อมต่อมอเตอร์กระแสตรงอื่นๆ (M2) ระหว่างพิน (OUT3 และ OUT4) ของ L293D
- เชื่อมต่อพิน (Vcc และกราวด์) ของ IR1 และ IR2 กับ +5 VDC และกราวด์ตามลำดับ
- ต่อพิน (OUT) ของ IR1 เข้ากับพิน (7) ของ Arduino Uno
- ต่อพิน (OUT) ของ IR2 เข้ากับพิน (6) ของ Arduino Uno
- เชื่อมต่อ Arduino Uni กับ Raspberry Pi ผ่าน USB

รูปที่ 7.7แสดงแผนภาพวงจรสำหรับหุ่นยนต์เดินตามเส้น

7.8.1 สูตร

```
นำเข้า pyrmata
นำเข้าเวลารอ
บอร์ด = pyrmata.Arduino('/dev/ttyUSB10')
ir1_pin = board.get_pin('d:7:i') # เชื่อมต่อเซ็นเซอร์ IR1 กับพิน 7 และใช้
    เป็นอินพุต
ir2_pin = board.get_pin('d:6:i') # เชื่อมต่อเซ็นเซอร์ IR2 กับพิน 6 และใช้
    เป็นอินพุต
```

```
M11_pin = board.get_pin('d:5:o') # ต่อขามอเตอร์ตัวที 5 แล้วใช้
    เป็นผลผลิต
M12_pin = board.get_pin('d:4:o') # ต่อขามอเตอร์ตัวที 4 แล้วใช้
    เป็นผลผลิต
M21_pin = board.get_pin('d:3:o') # ต่อขามอเตอร์ตัวทีสองกับ 3 และ
    ใช้เป็นผลผลิต
M22_pin = board.get_pin('d:2:o') # ต่อขามอเตอร์ตัวทีสองกับ 2 และ
    ใช้เป็นผลผลิต
มัน = pyrmata.util.Iterator(บอร์ด) # ใช้ตัววนซ้ำ
it.start() # เริ่มตัววนซ้ำ
ir1_pin.enable_reporting() # เปิดใช้งานการรายงานของเซ็นเซอร์ IR1
ir2_pin.enable_reporting() # เปิดใช้งานการรายงานของเซ็นเซอร์ IR2 ใน
ขณะที True:
                    ir1_state = ir1_pin.read () # อ่านเซ็นเซอร์ IR 1
                    ir2_state = ir2_pin.read () # อ่านเซ็นเซอร์ IR 2
                    ถ้า ir1_state == เท็จและ ir2_state == เท็จ:
                            M11_pin.write(1) # ทำ pin5 เป็น
                                สูง
                            M12_pin.write(0) # ทำ pin4 เป็น
                                ต่ำ
                            M21_pin.write(1) # ทำ pin3 เป็น
                                สูง
                        M22_pin.write(0) # make pin2 to LOW
                        print('forward') # print on terminal
                            wait.sleep(0.5) # ล่าช้า 500mSec
                    elif ir1_state == False และ ir2_state == True:
                            M11_pin.write(1) # ทำ pin5 เป็น
                                สูง
                            M12_pin.write(0) # ทำ pin4 เป็น
                                ต่ำ
                            M21_pin.write(0) # ทำ pin3 เป็น
                                ต่ำ
                            M22_pin.write(0) # ทำ pin2 เป็น
                                ต่ำ
                        พิมพ์ ('ซ้าย') # พิมพ์บนเทอร์มินัล
                        time.sleep (0.5) # ล่าช้า 500mSec
```

```
elif ir1_state == จริง และ ir2_state == เท็จ:
        M11_pin.write(0) # ทำ pin5 เป็นต่ำ
        M12_pin.write(0) # ทำ pin4 เป็นต่ำ
        M21_pin.write(1) # ทำ pin3 เป็นสูง
        M22_pin.write(0) # ทำ pin2 เป็นต่ำ
        พิมพ์ ('ถูกต้อง') # พิมพ์บนเทอร์มินัล
        time.sleep (0.5) # ล่าช้า 500mSec
elif ir1_state == True และ ir2_state == True:
        M11_pin.write(0) # ทำ pin5 เป็นต่ำ
        M12_pin.write(0) # ทำ pin4 เป็นต่ำ
        M21_pin.write(0) # ทำ pin3 เป็นต่ำ
        M22_pin.write(0) # ทำ pin2 เป็นต่ำ
    พิมพ์ ('หยุด') # พิมพ์บนเทอร์มินัล
    time.sleep (0.5) # ล่าช้า 500mSec
```