

7.2 การควบคุม Arduino ด้วย Python

ข้อต่อ "USB มาตรฐาน A" ใช้สำหรับเชื่อมต่อ Arduino กับ Raspberry Pi ตรวจสอบที่อยู่ USB ของ Arduino โดยเรียกใช้ "ls

- lrt /dev/tty*”ใน Raspberry Pi ของฉันมีรายการเป็น /dev/ttyUSB0 (จำค่านี้ไว้ใช้ภายหลัง)

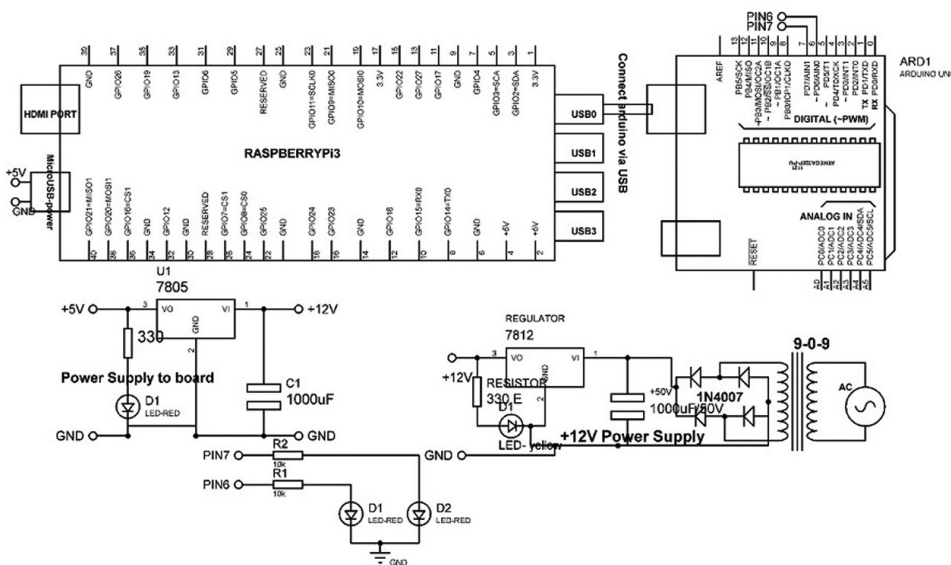
นำเข้าคลาส Arduino และ util จากโมดูล Pyrmata เพื่อควบคุม Arduino จากสคริปต์ Python บน Raspberry Pi หลังจากนั้น ให้สร้างวัตถุที่พบในขั้นตอนก่อนหน้านี้โดยใช้ที่อยู่ USB

> > > จาก pyrmata นำเข้า Arduino util

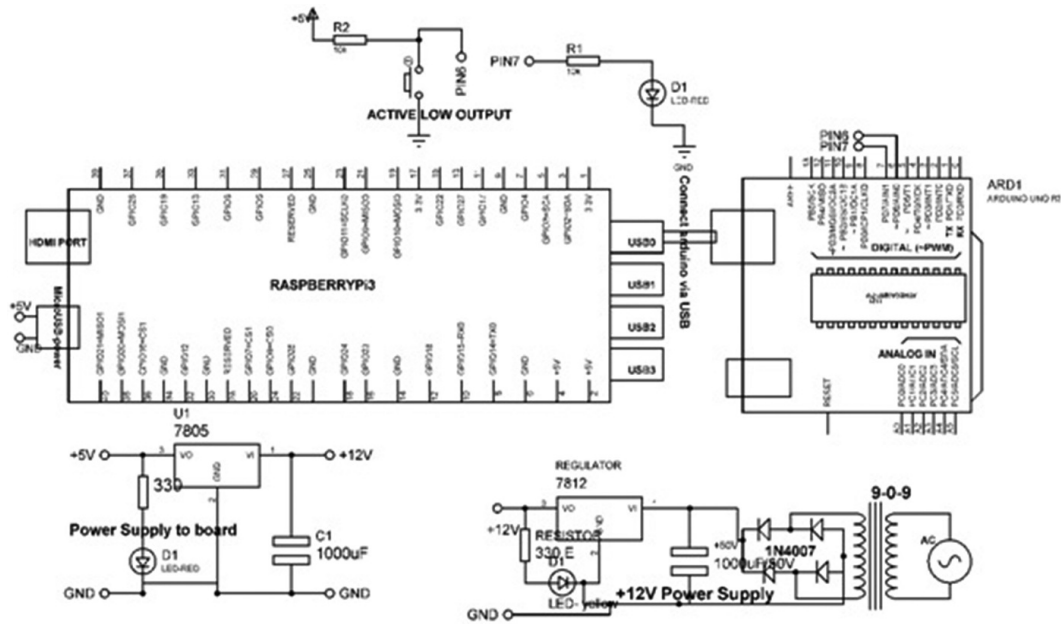
```
> > > u0 = Arduino('/dev/ttyUSB0')
```

7.3 เล่นกับ LED

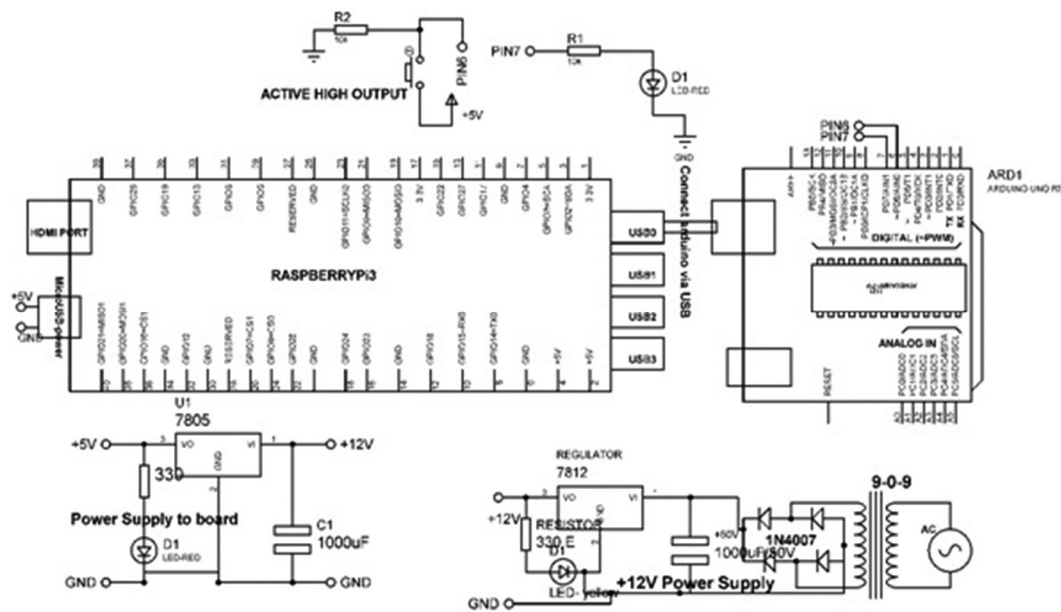
วัตถุประสงค์ของโครงการนี้คือการควบคุมเอาต์พุตดิจิทัลของ Arduino ผ่าน Raspberry Pi ด้วย Python ในการสร้างโปรเจกต์นี้ ให้เชื่อมต่อ LED กับพินดิจิทัลของ Arduino และเขียนโปรแกรม Python สั้นๆ เพื่อให้กะพริบ [รูปที่ 7.1](#) แสดงแผนภาพวงจรสำหรับการเชื่อมต่อของ LED ระบบประกอบด้วย Raspberry Pi3, Arduino Uno, แหล่งจ่ายไฟและไฟ LED สองดวงที่เชื่อมต่อกับ Pin6 และ Pin7 ของ Arduino โปรแกรมถูกเขียนขึ้นเพื่อให้ไฟ LED กะพริบหลังจากหนึ่งเวลาประมาณหนึ่ง



รูปที่ 7.1
แผนภาพวงจรสำหรับเชื่อมต่อ LED



รูป 7.2
การจัดเรียงแบบตึกลงสำหรับปุ่มอ่าน



รูป 7.3
การจัดเรียงแบบตึกขึ้นสำหรับปู่อ่าน

หลังจากนี้เปิดใช้งานพินโดยใช้คำสั่งต่อไปนี้

```
switch_pin.enable_reporting()
```

ไม่สามารถหยุดฟังก์ชันตัววนซ้ำได้ ดังนั้นเมื่อกด Ctrl+Z เพื่อออกจากหน้าต่าง ฟังก์ชันนั้นจะไม่มีอยู่

หากต้องการหยุดฟังก์ชันนี้ ให้ยกเลิกการเชื่อมต่อ Arduino จาก Raspberry Pi หรือเปิดหน้าต่างเทอร์มินัลอื่นแล้วใช้คำสั่ง kill:

```
$ sudo killall python
```

7.4.1 สูตรการอ่านการจัดวางแบบตึกลง

```
นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาที่รอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # ดene COM port ของ Arduino
button_pin = board.get_pin('d:6:i') # ดene pin 6 เป็นอินพุต led_pin =
board.get_pin('d:7:o') # ดene 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 == True: # 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
    else:
        print('ไม่ได้ออก') # พิมพ์สตริงบนเทอร์มินัล Pi
        led_pin.write (0) # เขียน '0' บนพิน 7
        wait.sleep(0.2) # ล่าช้า 0.2 วินาที
```

7.4.2 สูตรการอ่านการจัดวางแบบตึกขึ้น

```
นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาที่รอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # ดene 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 uu Pi terminal
```

```
led_pin.write(1) # ทำพิน 7 ถึง '1'  
wait.sleep(0.2) # ล่าช้า 0.2 วินาที
```

อื่น

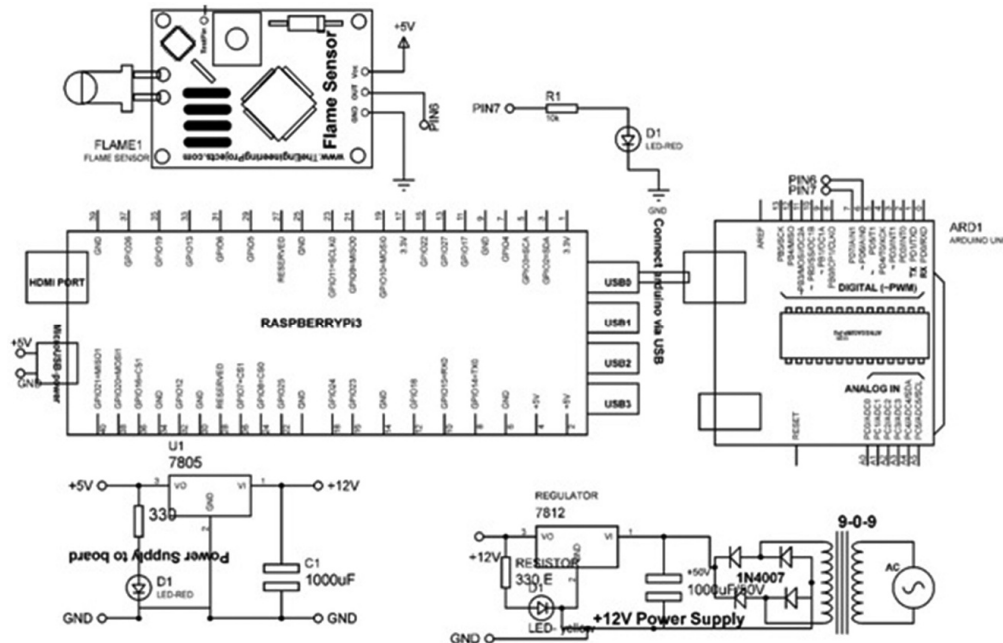
```

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

```

7.5 การอ่านเซ็นเซอร์เพลวไฟด้วย Pyfirmata

วัตถุประสงค์คือเพื่ออ่านเซ็นเซอร์ ame เป็นอินพุตด้วย Python บน Raspberry Pi เซ็นเซอร์ ame สามารถตรวจจับแสงอินฟราเรดที่มีความยาวคลื่นตั้งแต่ 700 ถึง 1,000 นาโนเมตร โพรบ ame อินฟราเรดไกลจะแปลงแสงที่ตรวจจับได้ในรูปของแสงอินฟราเรดให้เป็นกระแส มีแรงดันไฟฟ้าทำงาน 3.3 ถึง 5.2 V DC พร้อมเอาต์พุตดิจิทัลเพื่อบ่งชี้ว่ามีสัญญาณอยู่ ตัวเปรียบเทียบกับอนบอร์ด LM393 ใช้สำหรับการตรวจจับสภาพ เชื่อมต่อส่วนประกอบตามที่แสดงในรูปที่ 7.4 และตรวจสอบการทำงานโดยอัลโหลดสูตรที่อธิบายไว้ในมาตรา 7.5.1.



รูป 7.4
แผนภาพวงจรสำหรับการเชื่อมต่อเซ็นเซอร์ ame

7.5.1 โปรแกรมสำหรับการอ่านเซ็นเซอร์เปลวไฟ "ต่ำ" ที่ใช้งานอยู่

```
นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า
เวลาที่รอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # ดึง COMport of Arduino
ame_pin = board.get_pin('d:6:i') # กำหนด pin 6 เป็นอินพุตดิจิทัล
indicator_pin = board.get_pin('d:7:o') # กำหนด pin 7 เป็นเอาต์พุตดิจิทัล
pyrmata.util.Iterator (บอร์ด) # ใช้ตัววนซ้ำ
it.start() # start iterator
ame_pin.enable_reporting() # เปิดใช้งานอินพุตใน
ขณะที่ True: # infinite loop
    ame_state = ame_pin.read() # อ่านอินพุตดิจิทัล
    if ame_state == 1: # ตรวจสอบเงื่อนไข
        พิมพ์('ไม่มีอุปสรรค') # พิมพ์สตริงบน Pi Terminal
        indicator_pin.write(1) # เขียน '1' บน pin7 wait.sleep
        (0.2) # สLEEPเป็นเวลา 0.2 วินาที
    else:
        พิมพ์('พบสิ่งกีดขวาง') # พิมพ์สตริงบน Pi Terminal
        indicator_pin.write(0) # เขียน '0' บน pin7
        wait.sleep(0.2) # sleep สำหรับ 0.2 วินาที
```

7.6 การอ่านอินพุตแบบอะนาล็อกด้วย Pyfirmata

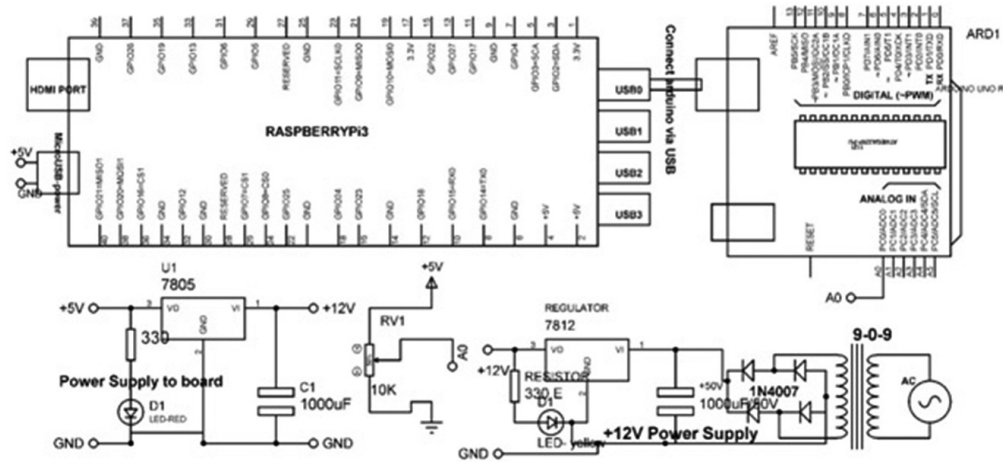
โพเทนชิออมิเตอร์ใช้เพื่อสาธิตการทำงานของเซ็นเซอร์แอนะล็อกด้วย Pyrmata เชื่อมต่อกับขา A0 ของ Arduino ([รูปที่ 7.5](#)). เมื่อพินได้รับการเชื่อมต่อเป็นพินอินพุตแบบอะนาล็อกในโปรแกรม จะเริ่มส่งค่าอินพุตไปยังพอร์ตอนุกรม หากไม่สามารถจัดการข้อมูลได้อย่างเหมาะสม ข้อมูลจะเริ่มบัฟเฟอร์ที่พอร์ตอนุกรมและโอเวอร์โวลต์อย่างรวดเร็ว สถานการณ์นี้สามารถจัดการกับโปรแกรมได้

ไลบรารี Pyrmata มีวิธีการรายงานและตัววนซ้ำเพื่อเอาชนะสถานการณ์นี้ `ti.enable_reporting()` เมรอดใช้เพื่อตั้งค่าพินอินพุตเพื่อเริ่มการรายงาน วิธีนี้ใช้ก่อนดำเนินการอ่านบนพิน:

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

เมื่อดำเนินการอ่านเสร็จแล้ว หมุดจะถูกตั้งค่าให้ปิดใช้งานการรายงาน:

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

រូបភាព 7.5

แผนภาพวงจรสำหรับการเชื่อมต่อโพเทนชิออมิเตอร์

หากต้องการอ่านพินแบบอะนาล็อก iterator หรือ thread ใช้ในรูปหลัก
 คลาสนี้ถูกปฏิเสธในโมดูล util ของแพ็คเกจ Pyrmata และนำเข้าก่อนที่จะนำไปใช้ในโค้ด:

จาก pyrmata นำเข้า Arduino util

การตั้งค่าพอร์ตบอร์ด

```
Arduino = 'COM3'
```

ບອຣ໌ດ = Arduino (ພອຣ໌ຕ)

นอน(5)

```
it = util.Iterator(board) # Start Iterator เพื่อหลีกเลี่ยงการ overflow
```

```
it.start()
```

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

7.6.1 សុពលភាព

นำเข้า pyrmata # นำเข้าไลบรารีของ pyrmata นำเข้า

เวลาที่รอ # นำเข้าไลบรารีของเวลา

```
board = pyrmata.Arduino('/dev/ttyUSB0') # ปลายเสพอร์ต COM ของ Arduino
```

```
POT_pin = board.get_pin('a:0:i') # กำหนดพิน A0 เป็นอินพุต =
```

```
pyrmata.util.Iterator (បាញ់) # ប្រើ iterator
```

```
it.start() # start iterator
```

POT_pin.enable_reporting() # เปิดใช้งานพินใน

ขณะที่ True: # infinite loop

```
POT_reading = POT_pin.read () # อ่านพินอะนาล็อก
```



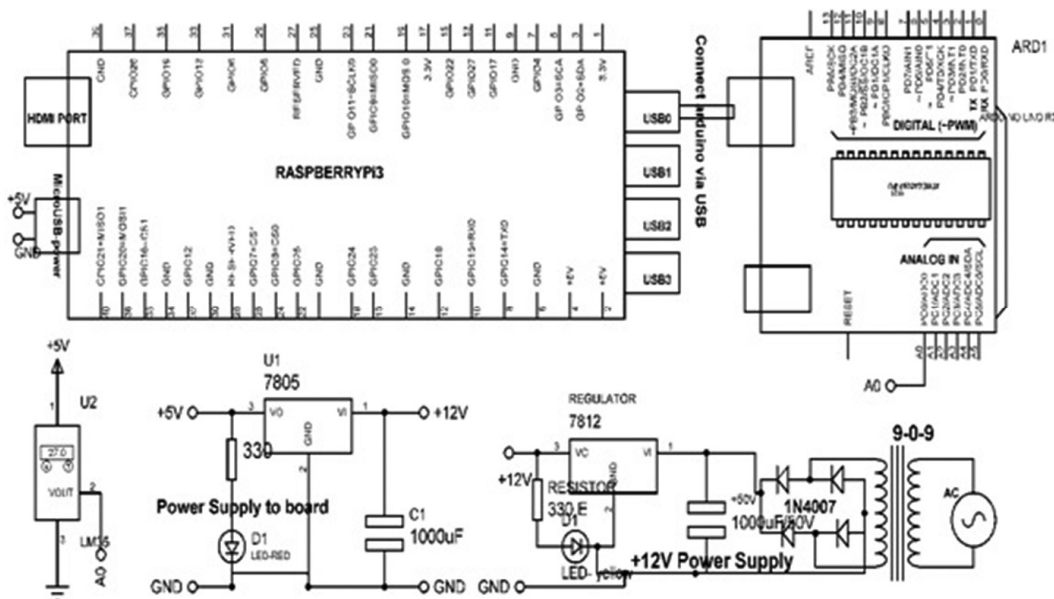
```
if POT_reading != None: # ตรวจสอบเงื่อนไข
    POT_voltage = POT_reading * 5.0 # แปลงระดับเป็น
    แรงดันไฟฟ้า
    พิมพ์("POT_reading=%f\t POT_voltage=%f"% (POT_
    การอ่าน POT_voltage))
    # printvalues บน Pi terminal
    wait.sleep(1) # sleep for 1 sec
```

อื่น:

```
print("ไม่มีการอ่าน") # พิมพ์สตริงบน Pi terminal
wait.sleep(1)# sleep for 1 sec
```

7.7 การอ่านเซ็นเซอร์อุณหภูมิด้วย Pyfirmata

เซ็นเซอร์อุณหภูมิซีรีส์ LM35 มีแรงดันเอาต์พุตตามสัดส่วนเชิงเส้นกับอุณหภูมิเซนติเกรด อุปกรณ์ LM35 ไม่ต้องการการสอบเทียบหรือการตัดแต่งเพื่อให้ความแม่นยำของ $\pm 1/4^{\circ}\text{C}$ ที่อุณหภูมิห้องและมีช่วงการตรวจวัดที่ -55°C ถึง 150°C . อุปกรณ์ LM35 ดึงกระแส 60 μA จากแหล่งจ่าย อุปกรณ์ซีรีส์ LM35 มีจำหน่ายในแพ็คเกจทรานซิสเตอร์ TO แบบสัญญาณภาค ในขณะที่ LM35C, LM35CA และ LM35D มีอยู่ในแพ็คเกจทรานซิสเตอร์ TO-92 แบบพลาสติก **รูปที่ 7.6** แสดงแผนภาพวงจรของการเชื่อมต่อ LM35 เอาต์พุตของ LM35 เชื่อมต่อกับพิน A0 ของ Arduino



รูป 7.6
แผนภาพวงจรของการเชื่อมต่อ LM35

7.7.1 សូតរ

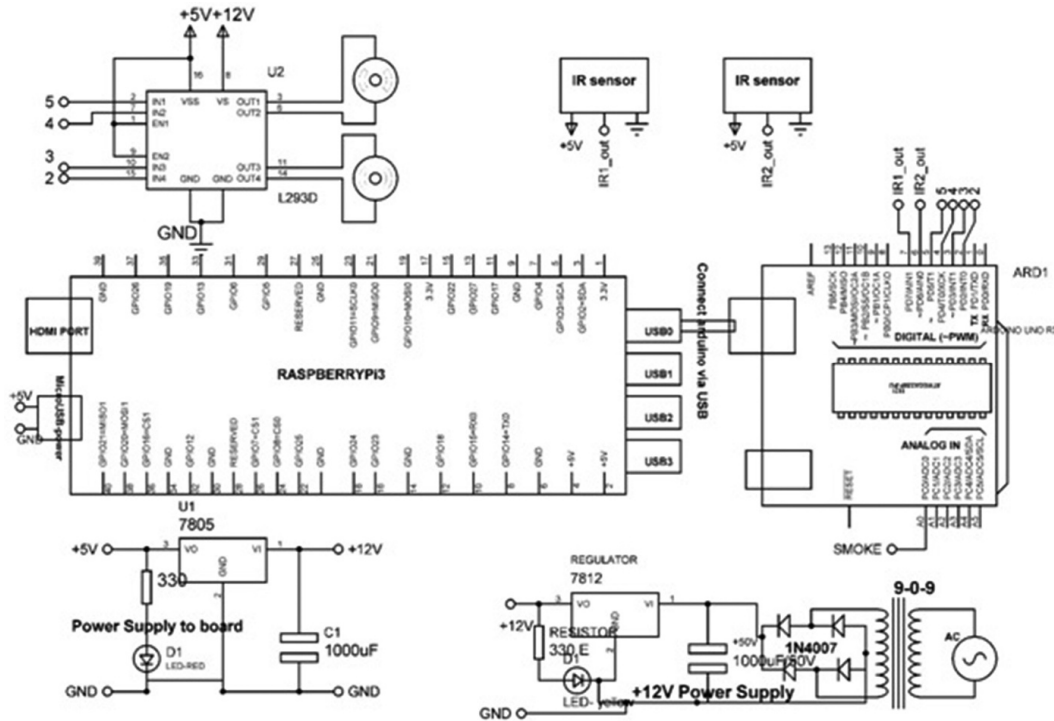
```
นำเข้า pyrmata # นำเข้าไลบรารีของเวลานำเข้า
pyrmata เนื่องจากรอ # นำเข้าไลบรารีของเวลา
board = pyrmata.Arduino('/dev/ttyUSB0') # ปลายเสพอร์ต COM ของ
Arduino
POT_pin = board.get_pin('a:0:i') # กำหนดพิน A0 เป็นอินพุต =
pyrmata.util.Iterator (บอร์ด) # ใช้ iterator
it.start() # start iterator
POT_pin.enable_reporting()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

7.8 หุ่นยนต์เดินตามเส้นด้วย Pyfirmata

หุ่นยนต์ติดตามเส้นตามภาพบน oor หรือเพดาน โดยปกติ เส้นสายตาจะเป็นสีดำบนพื้นผิวสีขาว แม้ว่าเส้นสีขาวบนพื้นผิวสีดำก็เป็นไปได้เช่นกัน หุ่นยนต์ตัวตามสายการผลิตใช้ในอุตสาหกรรมการผลิตสำหรับกระบวนการอัตโนมัติ เป็นหนึ่งในหุ่นยนต์พื้นฐานที่สุดสำหรับผู้เริ่มต้น เพื่อให้เข้าใจการออกแบบหุ่นยนต์ด้วย Raspberry Pi และ Arduino Uno ระบบจึงประกอบด้วยตัวขับเคลื่อนมอเตอร์ L293D มอเตอร์ DC สองตัว ล้ออิสระ (เพื่อเชื่อมต่อที่ด้านหน้าของหุ่นยนต์) เซ็นเซอร์ IR สองตัว และกำลังไฟฟ้า จัดหา.

ការ​ប្រើប្រាស់​បន្ត​ទៀត៖

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



រូបភាព 7.7

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 Uno กับ Raspberry Pi ผ่าน USB

รูปที่ 7.7 แสดงแผนภาพวงจรสำหรับหุ่นยนต์เดินตามเส้น

7.8.1 សុពលភាព

ນຳເຂົ້າ pyrmata

นำเข้าเวลารอ

```
u0309 = 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') # ต่อขามอเตอร์ตัวที่ 5 แล้วใช้
เป็นผลพลิต
M12_pin = board.get_pin('d:4:o') # ต่อขามอเตอร์ตัวที่ 4 แล้วใช้
เป็นผลพลิต
M21_pin = board.get_pin('d:3:o') # ต่อขามอเตอร์ตัวที่สองกับ 3 และ
ใช้เป็นผลพลิต
M22_pin = board.get_pin('d:2:o') # ต่อขามอเตอร์ตัวที่สองกับ 2 และ
ใช้เป็นผลพลิต
มัน = pyrmata.util.Iterator(บอร์ด) # ใช้ตัววนซ้ำ
it.start() # เริ่มตัววนซ้ำ
ir1_pin.enable_reporting() # เปิดใช้งานการรายงานของเซ็นเซอร์ IR1
ir2_pin.enable_reporting() # เปิดใช้งานการรายงานของเซ็นเซอร์ IR2 ใน
ขณะที่ 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 == จริ่ง และ ir2_state == เท็จ:
    M11_pin.write(0) # ทำ pin5 เป็น
        ต่ำ
    M12_pin.write(0) # ทำ pin4 เป็น
        ต่ำ
    M21_pin.write(1) # ทำ pin3 เป็น
        สูง
    M22_pin.write(0) # ทำ pin2 เป็น
        ต่ำ
    พิมพ์ ('ถูกต้อง') # พิมพ์บนเทอร์มินัล
    time.sleep (0.5) # ล่าช้า 500mSec
elif ir1_state == True และ ir2_state == True:
    M11_pin.write(0) # ทำ pin5 เป็น
        ต่ำ
    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 ('หยุด') # พิมพ์บนเทอร์มินัล
    time.sleep (0.5) # ล่าช้า 500mSec

```