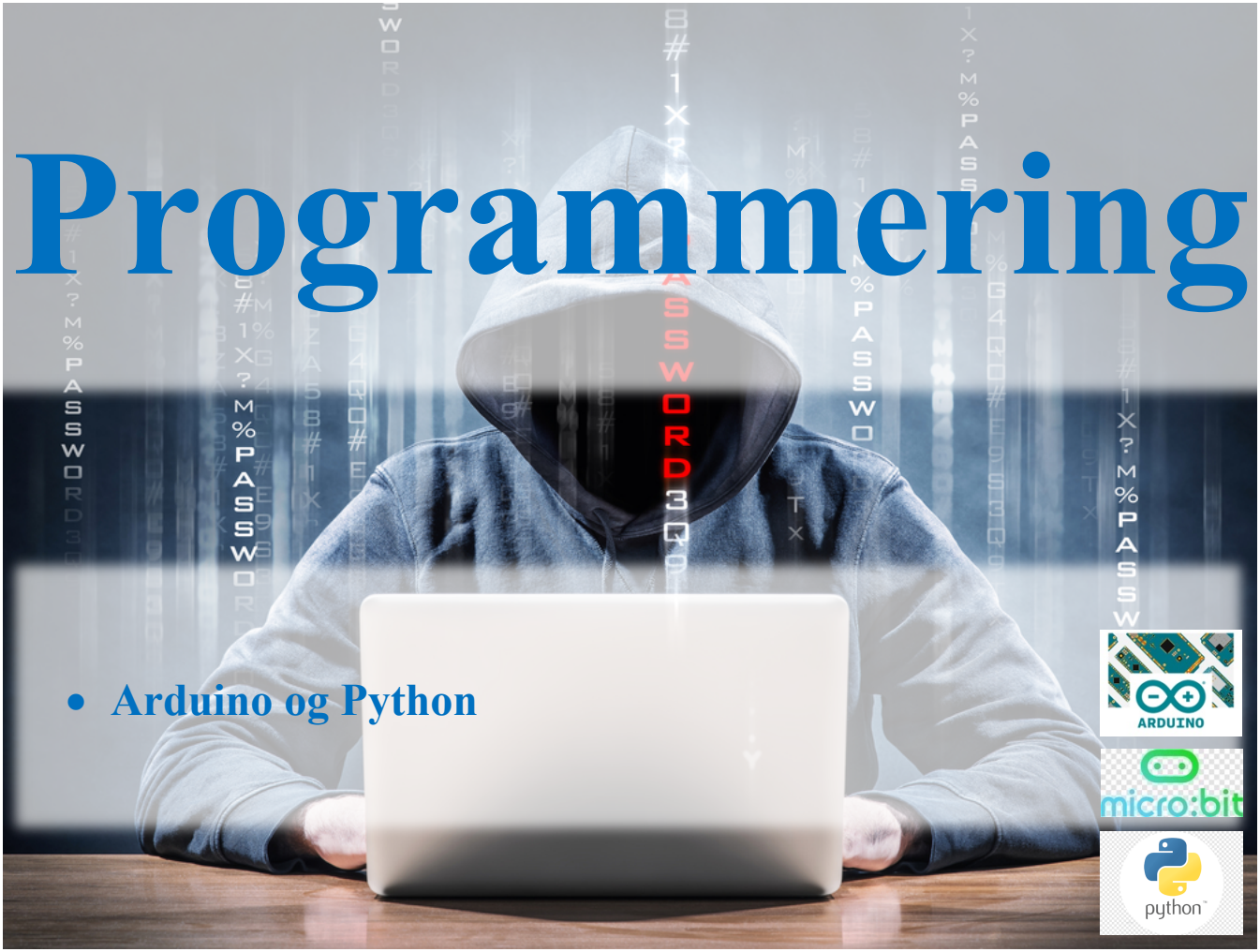
Programmering



- Arduino og Python

Arduino med Python

Python er i dag et av de mest brukte programmeringsspråkene og benyttes i dag av mange. Språket ble i sin tid utviklet av Guido Van Rossum i 1989, basert på at han ønsket et programmeringsspråk som var like enkelt å forstå som Engelsk. Dette enkle kurset er utarbeidet for å komme enkelt i gang med programmeringen og kunne se programmene i praksis ved hjelp av Arduino enheten.

I denne delen, Del 4 vil du få grunnleggende opplæring i Arduino med Python og lære:

1. Sette opp elektroniske kretser
2. Sett opp IDEL og Firmata-protokollen på Arduino for bruk av Python
3. Skriv grunnleggende applikasjoner for Arduino i Python
4. Kontrollere analoge og digitale innganger og utganger
5. Integrer Arduino-sensorer og -brytere med apper på høyere nivå
6. Utløse varsler på PCen og sende e-post ved hjelp av Arduino

Arduino-plattformen

Arduino er en åpen kildekode-plattform bestående av maskinvare og programvare som gjør det mulig for utvikling av interaktive elektronikkprosjekter. Fremveksten av Arduino trakk oppmerksomheten til fagfolk fra mange forskjellige bransjer, noe som bidro til starten av Maker Movement.

Med den økende populariteten til Maker Movement og konseptet med Tingenes Internett,har Arduino blitt en av de viktigste plattformene for elektronisk prototyping og utvikling.

Arduino bruker sitt eget programmeringsspråk, som ligner på C++. Det er imidlertid mulig å bruke Arduino med Python eller et annet programmeringsspråk på høyt nivå. Faktisk fungerer plattformer som Arduino godt med Python, spesielt for applikasjoner som krever integrasjon med sensorer og andre fysiske enheter.

Alt i alt kan Arduino og Python legge til rette for et effektivt læringsmiljø som oppfordrer utviklere til å begynne med elektronikkdesign. Hvis du allerede kjenner det grunnleggende om Python, vil du kunne komme raskere igang med Arduino ved å bruke Python til å kontrollere den.

Arduino-plattformen inkluderer både maskinvare- og programvareprodukter. I denne opplæringen bruker du Arduino-maskinvare og Python-programvare for å lære om grunnleggende kretser, samt digitale og analoge innganger og utganger.

Arduino Maskinvare

Hvis du vil teste eksemplene, må du montere kretsene ved å koble **til elektroniske komponenter**. Du trenger:

1. Et Arduino Uno eller et annet kompatibelt brett
2. Noen forskjellige komponenter, som listes opp for hver øvelse
3. Et koplingsbrett (Breadboard)
4. Jumper ledninger av ulike farger og størrelser

Disse komponenetene inngår i våre tidligere oppsett ut fra C++ og Arduino øvelsene.

Arduino Programvare

I tillegg til disse maskinvarekomponentene må du installere noe programvare. Plattformen inkluderer Arduino IDE, et integrert utviklingsmiljø for programmering av Arduino-enheter, blant andre nettbaserte verktøy.

Arduino ble designet for å gi deg mulighet til å programmere med mindre vanskelighetsgrad. Generelt følger du disse trinnene:

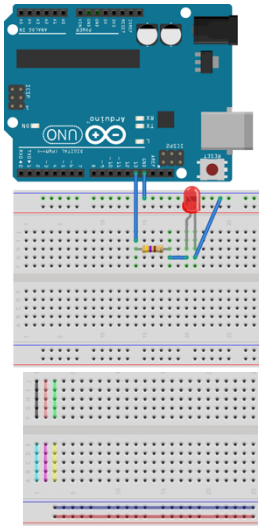
1. Koble kortet til PC-en
2. Installer og åpne Arduino IDE
3. Konfigurere innstillingene
4. Skrive koden
5. Trykk på en knapp på IDE for å laste opp programmet til enheten

Hvis du vil installere Arduino IDE på datamaskinen, laster du ned riktig versjon for operativsystemet fra Arduino-nettstedet. Se i dokumentasjonen for installasjonsinstruksjoner:

1. **Hvis du bruker Windows , brukerdu** Windows-installasjonsprogrammet til å sikre at du laster ned de nødvendige driverne for bruk av Arduino på Windows. Se Arduino-dokumentasjonen for mer informasjon.
2. **Hvis du bruker Linux**, må du kanskje legge til brukeren i enkelte grupper for å bruke den serielle porten til å programmere Arduino. Denne prosessen er beskrevet i Arduino installasjonsveiledning for Linux.
3. **Hvis du bruker macOS**, kan du installere Arduino IDE ved å følge Arduino installasjonsveiledning for OS X.

Merk:
Du skal bruke Arduino IDE i denne opplæringen, men Arduino har også en web editor som lar deg programmere Arduino ved hjelp av nettleseren.
Nå som du har installert Arduino IDE og samlet alle nødvendige komponenter, er du klar til å komme i gang med Arduino! Deretter laster du opp et "Hallo, verden!" -program til brettet ditt.

"Hallo, verden!" Med Arduino og Python



Tidligere har vi sett på «Hallo Verden», eller blink funksjonen- både internt og eksternt.

Tidligere har du lastet opp Blink-skissen til Arduino-brettet ditt. Arduino skisser er skrevet på et språk som ligner på C ++ og er kompilert og registrert på flash-minnet til mikrokontrolleren når du trykker Last *opp*. Mens du kan bruke et annet språk for å programmere Arduino mikrokontroller direkte, Python!

Det er imidlertid noen endringer som må gjøres for å bruke Arduino med Python eller andre språk. En idé er å kjøre hovedprogrammet på en PC og bruke den serielle tilkoblingen til å kommunisere med Arduino gjennom USB-kabelen. Skissen ville da være ansvarlig for å lese inngangene, sende informasjonen til PCen, og få oppdateringer fra PCen for å oppdatere Arduino-utgangene.

For å kontrollere Arduino fra PCen må du utforme en protokoll for kommunikasjon mellom PC og Arduino. Du kan for eksempel vurdere en protokoll med meldinger som følgende:

1. **VERDIEN AV PIN 13 ER HØY:** brukes til å fortelle PCen om statusen til digitale inngangspinner
2. **SETT PIN 11 LOW:** brukes til å be Arduino om å sette en port i nevnte tilstand

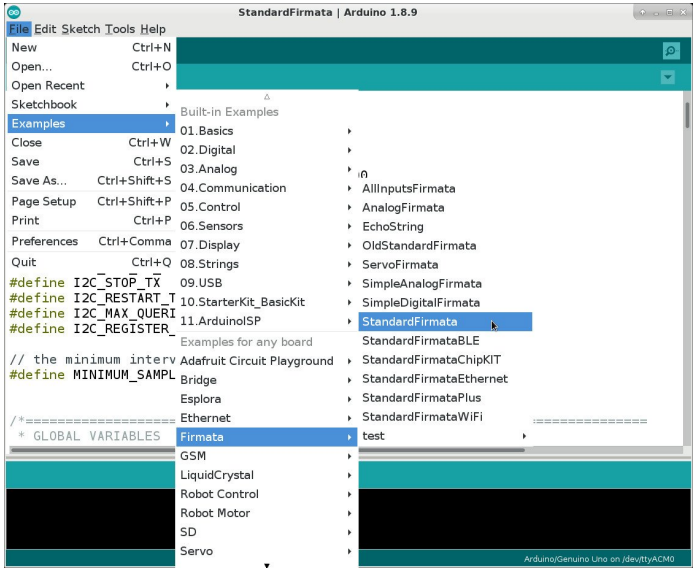
Når protokollen er definert, kan du skrive en Arduino «skisse» for å sende meldinger til PCen og oppdatere tilstander av pinnene i henhold til protokollen. På PCen kan du skrive et program for å kontrollere Arduino gjennom en seriell tilkobling, basert på protokollen du har designet. For dette kan du bruke hvilket språk og bibliotek du foretrekker, for eksempel Python og PySerial-biblioteket.

Heldigvis er det standard protokoller for å gjøre alt dette! **Firmata** er en av dem. Denne protokollen etablerer et seriell kommunikasjonsformat som lar deg lese digitale og analoge innganger, samt sende informasjon til digitale og analoge utganger.

Arduino IDE inneholder ferdige skisser som vil drive Arduino gjennom Python med Firmata-protokollen. På PC-siden er det implementeringer av protokollen på flere språk, inkludert Python. For å komme i gang med Firmata, la oss bruke den til å implementere et "Hallo, verden!" programet som en oppstart.

Laste opp Firmata-skissen

Før du skriver Python-programmet for å kjøre Arduino, må du laste opp Firmata-skissen slik at du kan bruke den protokollen til å styre brettet. Skissen er tilgjengelig i Arduino IDEs innebygde eksempler. Hvis du vil åpne den, åpner *du* Fil-menyen og *deretter Eksempler*, etterfulgt av *Firmata* og til slutt *StandardFirmata*:



Skissen lastes inn i et nytt IDE-vindu. For å laste den opp til Arduino, kan du følge de samme trinnene du gjorde før:

1. Koble USB-kabelen til PC-en.
2. Velg riktig bord og port på IDE.
3. Trykk Last *opp*.

Last ned Python IDLE

Installer IDLE på maskinen din, på skrivebord.

- Las ned link
- IDLE: Integrated Development and Learning Environment.

MERK: Selv om datamaskinen din opererer på 64-biters, kan du bruke 32-biters Python installasjon, på grunn av mangel på kompatibilitet med Arduino bibliotekene.

Last ned pyserial

PySerial er en Python API-modul som brukes til å lese og skrive serielle data til Arduino eller andre Microcontroller. For å installere på Windows, følg linken PySerials nedlastingsside og følg installasjonsbeskrivelsen :

1.Last ned PySerial fra lenken ovenfor.

2.Installer den ved å beholde innstillingen som standard Du bør være sikker på at Pyserial fungerte riktig.

For å gjøre dette; Du skriver inn :

```
importere serie
```

hvis du ikke har fått opp noen feil, så er du klar , hvis du får opp feil må vi sjekke installasjonen og Python IDLE forlengelse.

Etter at opplastingen er ferdig, vil du ikke merke noen aktivitet på Arduino. For å kontrollere det, trenger du fortsatt et program som kan kommunisere gjennom den serielle tilkoblingen. For å jobbe med Firmata-protokollen i Python, trenger du pyFirmata-pakken, som du kan installere med pip:

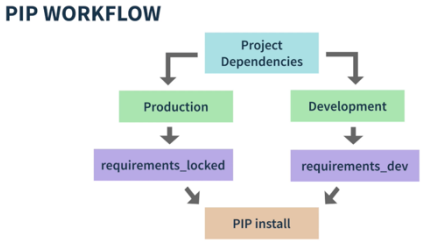
```
$ pip installere pyfirmata
```

Hva er pip? pip er standard pakkebehandling for Python. Den lar deg installere og administrere flere pakker som ikke er en del av Python **standardbibliotek**. Dette er en introduksjon til pip for nye Pythonistas.

Du lærer:

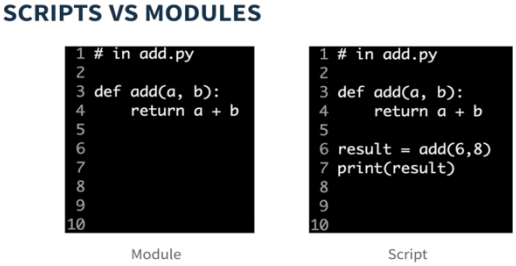
1. Installere tilleggspakker som ikke følger med standard Python-distribusjon
2. Finne pakker publisert til Python Package Index (PyPI)
3. Behandle krav til skript og programmer
4. Avinstallere pakker og avhengigheter

Som du vil se, er Python-utviklerne veldig aktive og har laget noen fine alternativer til pip som du vil lære om senere.

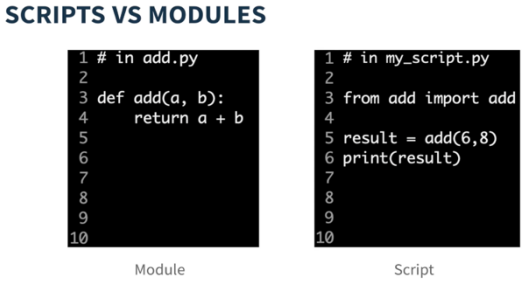


Python Module Index
Python bibliotek

Her ser vi typisk oppsett på script vs module. Bildet til høyre er en typisk Python script som beskriver handling og output- mens venstre del er en modul som kan importeres i andre programmer.



Her ser vi at vi importerer modulen inn i scripten til høyre.



Et **skript** er en Python-fil som er ment å kjøres direkte. Når du kjører den, bør den gjøre noe. Dette betyr at skript ofte vil inneholde kode skrevet utenfor omfanget av klasser eller funksjoner.

En **modul** er en Python-fil som er ment å importeres til skript eller andre moduler. Den definerer ofte medlemmer som klasser, funksjoner og variabler som er ment å brukes i andre filer som importerer den.

En **pakke** er en samling relaterte moduler som fungerer sammen for å gi visse funksjoner. Disse modulene finnes i en mappe og kan importeres akkurat som alle andre moduler. Denne mappen vil ofte inneholde en spesiell __init__ fil som forteller Python at det er en pakke, som potensielt inneholder flere moduler i undermapper

Det er ikke uvanlig at nye Pythonistas har problemer med å installere **pakker** og bruke **modulene** sine. Frustrerende feil som dette oppstår ofte, selv om du tror du har installert en pakke riktig:

```
>>>
ImportError: Ingen modul navngitt<package_name>
```

Dette skyldes at versjonen av Python du kjører skriptet med, ikke er konfigurert til å søke etter moduler der du har installert dem. Dette skjer når du bruker feil installasjon av pip for å installere pakker.
Generelt kommer hver Python-installasjon sammen med sin egen pip-kjørbare fil, som brukes til å installere pakker. Som standard vil den kjørbare pip-filen installere pakker på et sted der den spesifikke Python-installasjonen kan finne dem.

PIP pakker, Pypi org Last ned PyCharm

Installere pakker

Når installasjonen er fullført, kan du kjøre et tilsvarende Blink-program ved hjelp av Python og Firmata:

```
1       import pyfirmata
2       import time
3
4       board = pyfirmata.Arduino('/dev/ttyACM0')          // feltet I grønt er din port beskrivelse
5
6       while True:
7       board.digital[13].write(1)
8       time.sleep(1)
9       board.digital[13].write(0)
10              time.sleep(1)
```

Her er oversikt over hvordan dette programmet fungerer. Du importerer først pyfirmata til Arduinoen din. Denne bruker til å opprette en seriell forbindelse med Arduino-kortet, som representeres av kommandoen i linje 4. Du kan også konfigurere porten i denne linjen ved å sende et argument til pyfirmata. Arduino(). Du kan bruke Arduino IDE til å finne .

board.digital er en liste over de digitale pinnene til Arduino. Disse elementene har metodene lese() og skrive(), som vil lese og skrive tilstanden til pinnene. Som de fleste innebygde programmer, består dette programmet hovedsakelig av en uendelig løkke:

1. I linje 7 er digital pin 13 slått på, noe som slår på LED-lampen på i ett sekund.
2. I linje 8 settes inn en pause på 1 sek, 1000ms.
3. I linje 9 er denne pinnen slått av, som slår av LED-lampen i ett sekund.

Nå som du vet det grunnleggende om hvordan du kontrollerer en Arduino med Python, la oss gå gjennom noen programmer for å arbeide med innganger og utganger.

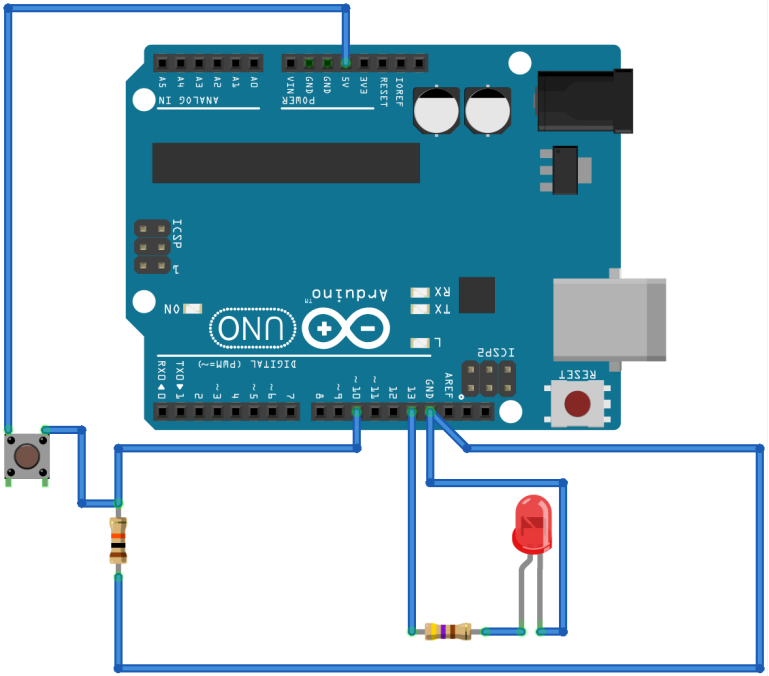
Lese digitale innganger

Digitale innganger kan bare ha to mulige verdier. I en krets er hver av disse verdiene representert av en annen spenning. Tabellen nedenfor viser den digitale inndatarepresentasjonen for et standard Arduino Uno-bord:

Verdi	Nivå	Spenning
0	Lav	0V (andre
1	Høy	5V. jeg har ikke noe å gjøre.

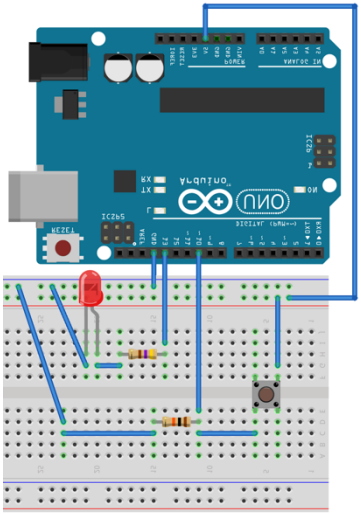
For å kontrollere LED-lampen bruker du en **trykknapp** til å sende digitale inngangsverdier til Arduino. Knappen skal sende 0V til brettet når den slippes og 5V til brettet når den trykkes. Figuren nedenfor viser hvordan du kobler knappen til Arduino-kortet:

Det kan hende du legger merke til at LED-lampen er koblet til Arduino på digital pin 13, akkurat som før. Digital pin 10 brukes som en digital inngang. For å koble til trykknappen må du bruke 10 KOhm-motstanden, som fungerer som en pull up/down i denne kretsen. En **nedtrekksmotstand** sikrer at den digitale inngangen får 0V når knappen slippes.



Når du slipper knappen, åpner du forbindelsen mellom de to ledningene på knappen. Siden det ikke flyter strøm gjennom motstanden, kobler pin 10 bare til bakken (GND).

Den digitale inngangen får 0V, som representerer **0** (eller **lav**)tilstand. Når du trykker på knappen, bruker du 5V på både motstanden og den digitale inngangen. En strøm strømmer gjennom motstanden og den digitale inngangen får 5V, som representerer **1** (eller **høy**)tilstand.



Du kan bruke et brødbrett til å montere kretsen ovenfor også:

Nå som du har montert kretsen, må du kjøre et program på PCen for å kontrollere den ved hjelp av Firmata. Dette programmet vil slå på LED, basert på tilstanden til trykknappen:

```
1       import pyfirmata
2       import time
3
4       board = pyfirmata.Arduino('/dev/ttyACM0')
5
6       it = pyfirmata.util.Iterator(board)
7       it.start()
8
9       board.digital[10].mode = pyfirmata.INPUT
10
```

```
11    while True:
12        sw = board.digital[10].read()
13    if sw is True:
14        board.digital[13].write(1)
15    else:
16        board.digital[13].write(0)
17        time.sleep(0.1)
```

1. Linje 1 og 2 importere pyfirmata og tid.
2. Linje 4 bruker pyfirmata. Arduino() for å sette forbindelsen med Arduino-brettet.
3. Linje 6 tilordner en iterator som skal brukes til å lese statusen for inngangene til kretsen.
4. Linje 7 starter iteratoren, som holder en løkke i parallell med hovedkoden. Sløyfen utfører board.iterate() for å oppdatere inngangsverdiene hentet fra Arduino-kortet.
5. Linje 9 angir pin 10 som en digital inngang med pyfirmata. INPUT. Dette er nødvendig siden standardkonfigurasjonen er å bruke digitale pinner som utganger.
6. Linje 11 starter en uendelig mens sløyfen. Denne løkken leser statusen for inndatapinnen, lagrer den i sw, og bruker denne verdien til åslå LED-lampen på eller av ved å endre verdien på pinne 13.
7. Linje 17 venter 0,1 sekunder mellom gjentagene av mens sløyfen. Dette er ikke strengt nødvendig, men det er et fint triks for å unngå overbelastning av CPU, som når 100% belastning når det ikke er en ventekommando i løkken.

pyfirmata tilbyr også en mer kompakt syntaks for å arbeide med inngangs- og utgangspinner. Dette kan være et godt alternativ for når du arbeider med flere pinner. Du kan skrive om det forrige programmet for å ha mer kompakt syntaks:

```
1   import pyfirmata
2   import time
3
4   board = pyfirmata.Arduino('/dev/ttyACM0')
5
6   it = pyfirmata.util.Iterator(board)
7   it.start()
8
9   digital_input = board.get_pin('d:10:i')
10  led = board.get_pin('d:13:o')
11
12  while True:
13      sw = digital_input.read()
14  if sw is True:
15      led.write(1)
16  else:
17          led.write(0)
18  time.sleep(0.1)
```

I denne versjonen bruker du board.get_pin() til å opprette to objekter. digital_input representerer den digitale inngangstilstanden, og led representerer LED-tilstanden. Når du kjører denne metoden, må du sende et strengargument som består av tre elementer atskilt med kolon:

1. **Typen stift** (en for analog eller d for digital)
2. **Nummeret på** stiften
3. **Modusen** på stiften (i for inngang eller o for utgang)

Siden digital_input er en digital inngang ved hjelp av pin 10, passerer du argumentet 'd:10:i'. LED-tilstanden er satt til en digital utgang ved hjelp av pin 13, så det ledede argumentet er 'd:13:o'.

Når du bruker board.get_pin(), er det ikke nødvendig å eksplisitt sette opp pin 10 som en inngang som du gjorde før med pyfirmata. INPUT. Når pinnene er angitt, kan du få tilgang til statusen til en digital inndatanål ved hjelp av read()og angi statusen for en digital utgangspinne med write().

Digitale innganger er mye brukt i elektronikkprosjekter. Flere sensorer gir digitale signaler, som tilstedeværelse eller dørsensorer, som kan brukes som innganger til kretsene dine. Det er imidlertid noen tilfeller der du må måle analoge verdier, for eksempel avstand eller fysiske

mengder. I neste avsnitt vil du se hvordan du leser analoge innganger ved hjelp av Arduino med Python.

Lese analoge innganger

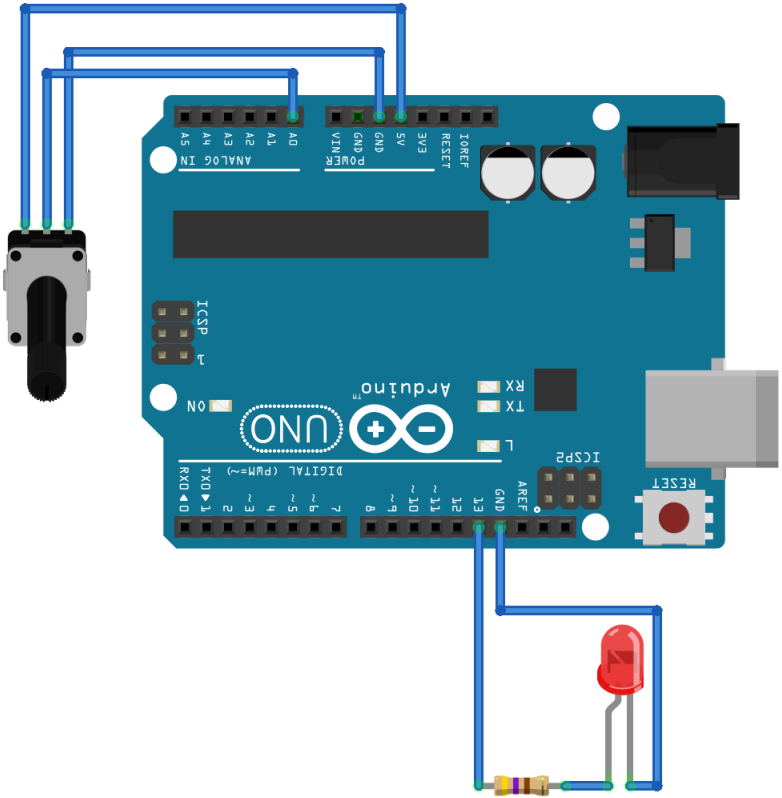
I motsetning til digitale innganger, som bare kan være på eller av, **brukes analoge** innganger til å lese verdier i enkelte områder. På Arduino Uno varierer spenningen til en analog inngang fra 0V til 5V. Passende sensorer brukes til å måle fysiske mengder, for eksempel avstander. Disse sensorene er ansvarlige for å koding disse fysiske mengdene i riktig spenningsområde, slik at de kan leses av Arduino.

For å lese en analog spenning bruker Arduino **en analog-til-digital omformer (ADC),**som konverterer inngangsspenningen til et digitalt nummer med et fast antall biter. Dette bestemmer oppløsningen av konverteringen. Arduino Uno bruker en 10-biters ADC og kan bestemme 1024 forskjellige spenningsnivåer.

Spenningsområdet for en analog inngang er kodet til tall fra 0 til 1023. Når 0V brukes, koder Arduino den til tallet **0**. Når 5V brukes, er det kodede nummeret **1023**. Alle mellomliggende spenningsverdier er proporsjonalt kodet.

Et potensiometer er en variabel motstand som du kan bruke til å stille inn spenningen som brukes på en Arduino analog inngang. Du kobler den til en analog inngang for å kontrollere frekvensen til en blinkende LED:

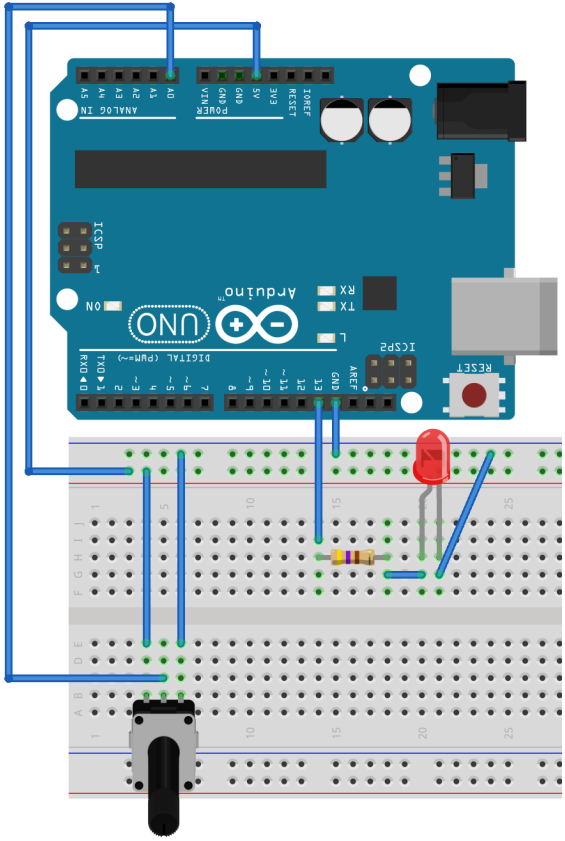
I denne kretsen er LED-lampen satt opp akkurat som før. Endeterminalene på potensiometeret er koblet til jording (GND) og 5V pinner. På denne måten kan sentralterminalen (markøren) ha spenning i 0V til 5V-området, avhengig av posisjonen, som er koblet til Arduino på analog pin A0.



Ved hjelp av et brødbrett kan du montere denne kretsen på følgende måte:

Før du kontrollerer LED-lampen, kan du bruke kretsen til å kontrollere de forskjellige verdiene Arduino leser, basert på potensiometerets posisjon. Hvis du vil gjøre dette, kjører du følgende program på PCen:

```
 1import pyfirmata
 2import time
 3
 4board = pyfirmata.Arduino('/dev/ttyACM0')
 5it = pyfirmata.util.Iterator(board)
 6it.start()
 7
 8analog_input = board.get_pin('a:0:i')
 9
10while True:
11    analog_value = analog_input.read()
12    print(analog_value)
13    time.sleep(0.1)
```



I linje 8 setter du opp analog_input som den analoge A0-inngangspinnen med argumentet 'a:0:i'. Inne i den uendelige mens løkken leser du denne verdien, lagrer den i analog_valueog viserutdataene til konsollen med print(). Når du flytter potensiometeret mens programmet kjører, bør du sende ut på samme måte som vist her:

```
0.0
0.0293
0.1056
0.1838
0.2717
0.3705
0.4428
0.5064
0.5797
0.6315
```

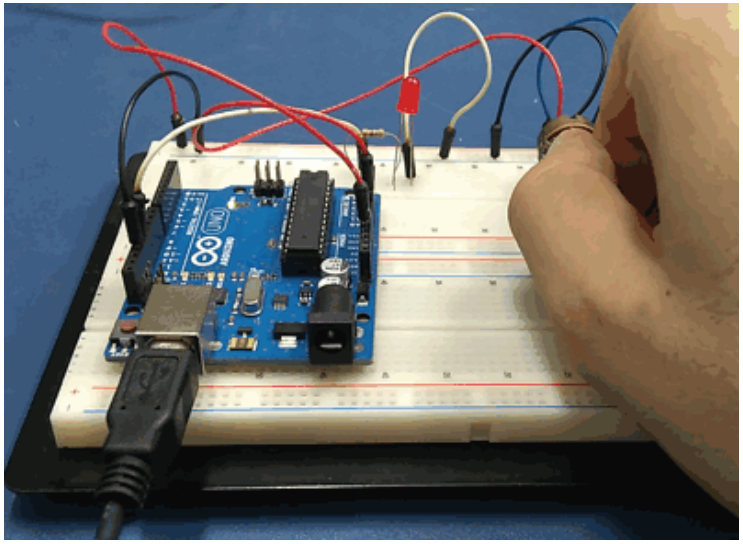
De trykte verdiene endres, fra 0 når posisjonen til potensiometeret er i den ene enden til 1 når den er i den andre enden. Vær oppmerksom på at dette er flottørverdier, som kan kreve konvertering avhengig av programmet.

Hvis du vil endre frekvensen til den blinkende lysdioden, kan du bruke analog_value til å kontrollere hvor lenge LED-lampen skal holdes på eller av:

```
1import pyfirmata
2import time
3
4board = pyfirmata.Arduino('/dev/ttyACM0')
5it = pyfirmata.util.Iterator(board)
6it.start()
7
8analog_input = board.get_pin('a:0:i')
9led = board.get_pin('d:13:o')
10
11while True:
12    analog_value = analog_input.read()
13    if analog_value is not None:
14        delay = analog_value + 0.01
15        led.write(1)
16        time.sleep(delay)
17        led.write(0)
18        time.sleep(delay)
19    else:
20        time.sleep(0.1)
```

Her beregner du forsinkelse som analog_value + 0,01 for å unngå å ha forsinkelse lik null. Ellers er det vanlig å få en analog_value av Ingen i løpet av de første iterasjonene. Hvis du vil unngå å få en feilmelding når du kjører programmet, bruker du en betinget linje 13 til å teste om analog_value er Ingen. Deretter kontrollerer du perioden for den blinkende LED-lampen.

Prøv å kjøre programmet og endre posisjonen til potensiometeret. Du vil legge merke til hyppigheten av de blinkende LED-endringene:



Nå har du sett hvordan du bruker digitale innganger, digitale utganger og analoge innganger på kretsene dine. I neste avsnitt ser du hvordan du bruker analoge utganger.

Bruke analoge utganger

I noen tilfeller er det nødvendig å ha en **analog utgang for å** drive en enhet som krever et analogt signal. Arduino inkluderer ikke en ekte analog utgang, en hvor spenningen kan settes til noen verdi i et bestemt område. Arduino inkluderer imidlertid flere **PULSE Width Modulation** (PWM) utganger.

PWM er en modulasjonsteknikk der en digital utgang brukes til å generere et signal med variabel kraft. For å gjøre dette bruker den et digitalt signal om konstant frekvens, der **driftssyklusen** endres i henhold til ønsket kraft. Driftssyklusen representerer brøkdelen av perioden der signalet er satt til **høy**.

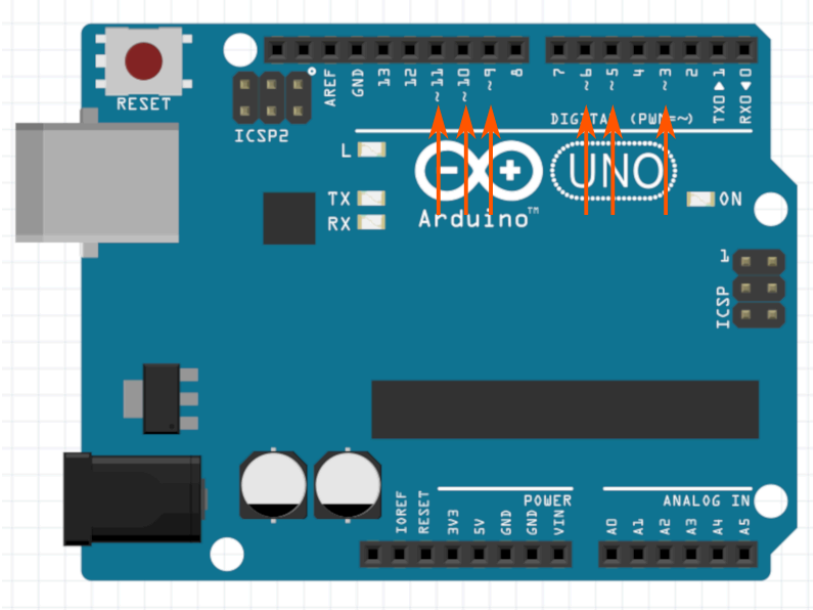
Ikke alle Arduino digitale pinner kan brukes som PWM utganger. De som kan identifiseres av en tilde (~):

Flere enheter er designet for å være drevet av PWM-signaler, inkludert noen motorer. Det er til og med mulig å få et ekte analogt signal fra PWM-signalet hvis du bruker analoge filtre. I det forrige eksemplet brukte du en digital utgang til å slå et LED-lys av eller på. I denne

delen bruker du PWM til å kontrollere lysstyrken på en LED, i henhold til verdien av en analog inngang gitt av et potensiometer.

Når et PWM-signal påføres en LED, varierer lysstyrken i henhold til driftssyklusen til PWM-signalet. Du kommer til å bruke følgende krets:

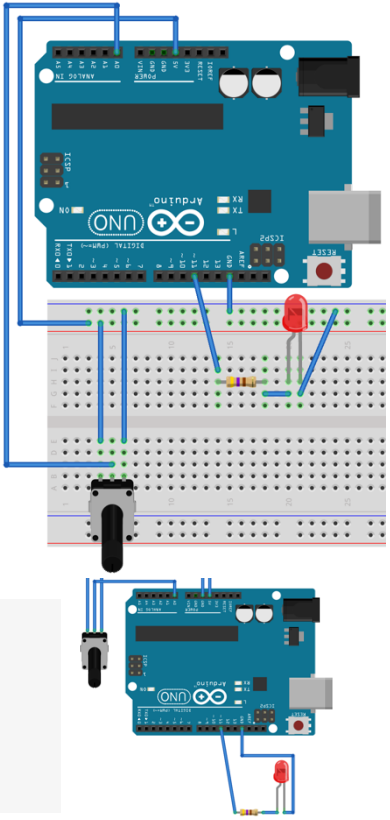
Denne kretsen er identisk med den som brukes i forrige avsnitt for å teste den analoge inngangen, med unntak av en forskjell. Siden det ikke er mulig å bruke PWM med pinne 13, er den digitale utgangspinnen som brukes til LED-lampen pin 11.



Du kan bruke et brødbrett til å montere kretsen på følgende måte:

Når kretsen er montert, kan du styre LED-lampen ved hjelp av PWM med følgende program:

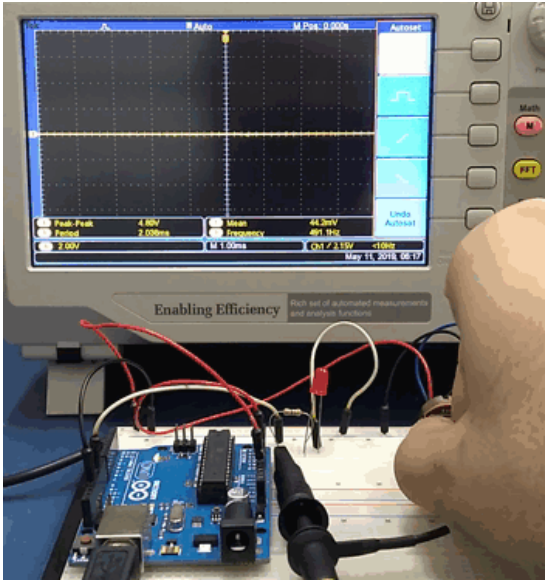
```
1import pyfirmata
2import time
3
4board = pyfirmata.Arduino('/dev/ttyACM0')
5
6it = pyfirmata.util.Iterator(board)
7it.start()
8
9analog_input = board.get_pin('a:0:i')
10led = board.get_pin('d:11:p')
11
12while True:
13    analog_value = analog_input.read()
14    if analog_value is not None:
15        led.write(analog_value)
16    time.sleep(0.1)
```



Det er noen få forskjeller fra det programmet du brukte tidligere:

1. **I linje 10,**kan du angi førte til PWM-modus ved å sende argumentet 'd:11:p'.
2. **I linje 15**ringer du led.write() med analog_value som et argument. Dette er en verdi mellom 0 og 1, lest fra den analoge inngangen.

Her kan du se LED-virkemåten når potensiometeret flyttes:



For å vise endringene i driftssyklusen, er et oscilloskop koblet til pin 11. Når potensiometeret er i nullposisjon, kan du se at LED-lampen er slått av, da pinne 11 har 0V på utgangen. Når du vrir potensiometeret, blir LED-lampen lysere etter hvert som PWM-driftssyklusen øker. Når du snur potensiometeret hele veien, når driftssyklusen 100%. Led-lampen slås kontinuerlig på ved maksimal lysstyrke.

Med dette eksemplet har du dekket det grunnleggende om å bruke en Arduino og dens digitale og analoge innganger og utganger. I neste del vil du se et program for bruk av Arduino med Python for å kjøre hendelser på PCen.

Bruke en sensor til å utløse et varsel

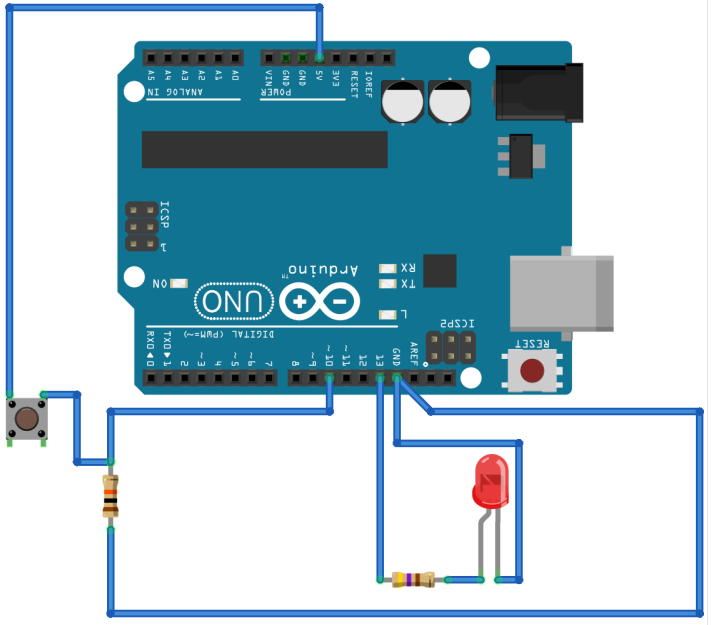
Firmata er en fin måte å komme i gang med Arduino med Python, men behovet for en PC eller annen enhet for å kjøre programmet kan være kostbart, og denne tilnærmingen kan ikke være praktisk i noen tilfeller. Men når det er nødvendig å samle inn data og sende den til en PC ved hjelp av eksterne sensorer, gjør Arduino og Firmata en god kombinasjon.

I denne delen bruker du en trykknapp som er koblet til Arduino, til å **etterligne en digital sensor**og utløse et varsel påmaskinen. For en mer praktisk applikasjon kan du tenke på trykknappen som en dørsensor som for eksempel utløser et alarmvarsel.

Hvis du vil vise varselet på PCen, skal du bruke Tkinter, standard Python GUI-verktøykasse. Dette viser en meldingsboks når du trykker på knappen. For en grundig intro til Tkinter, sjekk ut Python GUI Programmering Med Tkinter.

Du må montere den samme kretsen som du brukte i eksemplet på digital inngang:

Når du har montert kretsen, bruker du følgende program til å utløse varslene:



```
1import pyfirmata
2import time
3import tkinter
4from tkinter import messagebox
5
6root = tkinter.Tk()
7root.withdraw()
8
9board =
pyfirmata.Arduino('/dev/ttyACM0')
10
11it = pyfirmata.util.Iterator(board)
```

```
12it.start()
13
14digital_input = board.get_pin('d:10:i')
15led = board.get_pin('d:13:o')
16
17while True:
18    sw = digital_input.read()
19    if sw is True:
20        led.write(1)
21        messagebox.showinfo("Notification", "Button was pressed")
22        root.update()
23        led.write(0)
24    time.sleep(0.1)
```

This program is similar to the one used in the digital input example, with a few changes:

1. **Linje 3 og 4** importerer biblioteker som trengs for å konfigurere Tkinter.
2. **Linje 6** oppretter Tkinters hovedvindu.
3. **Linje 7** ber Tkinter om ikke å vise hovedvinduet på skjermen. I dette eksemplet trenger du bare å se meldingsboksen.

4. **Linje 17** starter mens-sløyfen:
    1. Når du trykker på knappen, slås LED-lampen på, og messagebox.showinfo()viser en meldingsboks.
    2. Løkken stopper på pause til brukeren trykker *på OK*. På denne måten forblir LED-lampen på så lenge meldingen er på skjermen.
    3. Når brukeren trykker *OK*, fjerner root.update() meldingsboksen fra skjermen og LED-lampen er slått av.

Hvis du vil utvide varslingseksemplet, kan du til og med bruke trykkknappen til å sende en e-post når du trykker på:

```
1import pyfirmata
2import time
3import smtplib
4import ssl
5
6def send_email():
7    port = 465  # For SSL
8    smtp_server = "smtp.gmail.com"
9    sender_email = "<your email address>"
10    receiver_email = "<destination email address>"
```

```
11    password = "<password>"
12    message = """Subject: Arduino Notification\n The switch was turned on."""
13
14    context = ssl.create_default_context()
15    with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
16        print("Sending email")
17        server.login(sender_email, password)
18        server.sendmail(sender_email, receiver_email, message)
19
20board = pyfirmata.Arduino('/dev/ttyACM0')
21
22it = pyfirmata.util.Iterator(board)
23it.start()
24
25digital_input = board.get_pin('d:10:i')
26
27while True:
28    sw = digital_input.read()
29    if sw is True:
30        send_email()
31        time.sleep(0.1)
```

Du kan lære mer om send_email() i Sende e-post med Python. Her kan du konfigurere funksjonen med legitimasjon for e-postserver, som skal brukes til å sende e-posten.

Merk: Hvis du bruker en Gmail-konto til å sende e-postene, må du aktivere alternativet *Tillat mindre sikre* apper. Hvis du vil ha mer informasjon om hvordan du gjør dette, kan du se Sende e-post med Python.

Med disse eksempelprogramene har du sett hvordan du bruker Firmata til å samhandle med mer komplekse Python-programmer. Firmata lar deg bruke en hvilken som helst sensor som er koblet til Arduino for å få data for ditt bruksområde. Deretter kan du behandle dataene og ta beslutninger i hovedprogrammet. Du kan til og med bruke Firmata til å sende data til Arduino-utganger, kontrollere brytere eller PWM-enheter.

Hvis du er interessert i å bruke Firmata til å samhandle med mer komplekse programmer, kan du prøve ut noen av disse prosjektene:

1. En temperaturmonitor som varsler deg når temperaturen blir for høy eller lav
2. En analog lyssensor som kan fornemme når en lyspære er utbrent
3. En vannsensor som automatisk kan slå på sprinkleranleggene når bakken er for tørr

Konklusjon

Mikrokontrollerplattformer er på vei opp, takket være den økende populariteten til Maker Movement og Tingenes Internett. Plattformer som **Arduino** får mye oppmerksomhet spesielt, da de tillater utviklere akkurat som deg å bruke sine ferdigheter og dykke inn i elektroniske prosjekter.

Du lærte å:

1. Utvikle applikasjoner med Arduino og Python
2. Bruke Firmata-protokollen
3. Kontrollere analoge og digitale innganger og utganger
4. Integrer sensorer med Python-programmer på høyere nivå

Du så også hvordan Firmata kan være et veldig interessant alternativ for prosjekter som krever en PC og er avhengig av sensordata. I tillegg er det en enkel måte å komme i gang med Arduino hvis du allerede kjenner Python!

Sjekkpunkter, hva har jeg lært?

I denne delen har vi gått gjennom noen eksempler på hvordan Arduino kan benyttes sammen med Python. Bruk av Firmata, Analoge og digitale signaler.

- ____ Sette opp Python sammen med Arduino
- ____ Bruk av Python sammen med Arduino i grunnleggende øvelser

https://create.arduino.cc/projecthub/smart-tech/programming-arduino-using-python-f3d2c0