**Aiuto multimetro**

|  |  |  |  |
| --- | --- | --- | --- |
| **Posizione del selettore** | **Grandezza misurata** | **Come si collega** |  |
| **V ⎓**  **(Volt)** | Tensione continua | In parallelo al circuito |
| **A ⎓  (Ampere)** | Corrente continua | In serie al circuito  Current Measurement Using Multimeter - ElectronicsHub |
| **Ω**  **(Ohm)** | Resistenza | Senza alimentazione |

**Esperimento 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Una pila | Due pile in serie | Due pile in parallelo | Portabatterie accesso | Portabatterie spento |
|  |  |  |  | |
| **Misura tensione** | **Misura tensione** | **Misura tensione** | **Misura tensione** | **Misura tensione** |
| V | V | V | V | V |

**Aiuto multimetro**

|  |  |  |  |
| --- | --- | --- | --- |
| **Posizione del selettore** | **Grandezza misurata** | **Come si collega** | Immagine che contiene dispositivo, metro, testo, elettronica  Descrizione generata automaticamente |
| **V ⎓**  **(Volt)** | Tensione continua | In parallelo al circuito |
| **A ⎓  (Ampere)** | Corrente continua | In serie al circuito |
| **Ω**  **(Ohm)** | Resistenza | Senza alimentazione  In parallelo |

**Aiuto breadboard**

Immagine che contiene Rettangolo, modello, schermata

Descrizione generata automaticamente

Le rette rappresentano le connessioni elettriche fra i vari fori della breadboard

**Esperimento 1 – motore**

Inserire graffette nella breadboard (1 e 2), poi infilare l’asse della bobina all’interno delle graffette (3), poi i magneti (4) e in fine i fili delle batterie nelle file verticali della breadboard (5).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4**  **5**  **5**  **3**  **3**  **1**  **2** | | 5. COLLEGARE BATTERIE - E +  3. INFILARE LE ESTREMITA’ DELLA BOBINA DENTRO LE GRAFFETTE  4. MAGNETI (ALMENO 3)  2. GRAFFETTA **+**  1. GRAFFETTA **–** | | |
| RISULTATO DA OTTENERE | |  | | |
| Segnare con la X il senso di rotazione osservato della bobina guardandola come sulla foto di sopra |  |  |  |  |

**Descrivete un modo per cambiare verso di rotazione e verificatelo**

**...............................................................................................................................**

**...............................................................................................................................**

**...............................................................................................................................**

**...............................................................................................................................**

**...............................................................................................................................**

**Esperimento 2 – misura resistenza**

Valore misurato : …………………… **Ω**

**Esperimento 3 – più resistenze**

|  |  |  |  |
| --- | --- | --- | --- |
| Resistenza 1 da sola | Resistenza 2 da sola | Due resistenze uguali in serie | Due resistenze uguali in parallelo |
|  | |  |  |
| **Misura R1** | **Misura R2** | **Misura R3** | **Misura R4** |
| …………………. **Ω** | ……………… **Ω** | …………… **Ω** | …………. **Ω** |

**Esperimento 4 – Python calcolatore**

Lanciare Thonny e eseguire nel REPL le istruzioni seguenti

|  |  |
| --- | --- |
| Istruzione | Risposta di Python |
| 1+1 | …………………………………………………………………. |
| 3\*10\*\*15 | ………………………………………………………………….. |

**Aiuto leggi di ohm e board**

|  |  |
| --- | --- |
| **TRIANGOLO**  Immagine che contiene triangolo, diagramma, design  Descrizione generata automaticamente | In python:  def ohm\_u(r, i):      return r \* i  print(ohm\_u(100, 0.1))  def ohm\_i(u, r):      return u / r  print(ohm\_i(4, 2000))  def ohm\_r(u, i):      return u / i  print(ohm\_r(4, 0.001)) |
| TOP_IMG | Pin |
| **BOARD PICO S2** | **SCHEMA CON NUMERO PINS** |

**Esperimento 1 – LED e misura di tensione**

|  |
| --- |
| Rappresentazione |
| Schema elettrico |

**Risultati sperimentali**

|  |  |  |  |
| --- | --- | --- | --- |
| Colore del diodo ? | ........................ | Tensione attorno al diodo? | U1 = .................. V |
| A Tensione attorno alla resistenza? | U2 = ............ V | Tensione delle batterie? | U3 = ................. V |
| A cosa è uguale U1 + U2 ? | ............. | *Abbiamo misurato il giorno 1 la tensione batterie* | |

Calcolo della corrente applicando la legge di Ohm alla resistenza: ......................... A

**Esperimento 2 – Board e Micropython**

|  |  |
| --- | --- |
| **Circuito** | **Istruzioni** |
| Immagine che contiene circuito, elettronica, Ingegneria elettronica, Componente elettrico  Descrizione generata automaticamente | Collegare il cavo USB alla board  Collegare Thonny alla board  Eseguire nel REPL di Thonny  Pin(10).on()  Pin(10).off() |

**Esperimento 1 – Pilotare lo schermo SSD**

|  |  |
| --- | --- |
| **Circuito** | **Istruzioni** |
|  | Collegare il cavo USB alla board  Collegare Thonny alla board  Aprire main.py sulla board  display.text('Ciao <nome>', 40, 12, 1)  display.show()  Salvare e RESET |

**Esperimento 2 – TouchPAD**

|  |  |
| --- | --- |
| **Circuito** | **Istruzioni di collegamento** |
| Immagine che contiene circuito, elettronica, Ingegneria elettronica, Componente elettrico  Descrizione generata automaticamente | Collegare la graffetta aperta al pin 4 della board usando la fila verticale della breadboard  Thonny / main.py  tp = TouchPad(Pin(4))  while True:      print(tp.read())      sleep\_ms(250) |

Quale valore soglia possiamo usare per riconoscere il tocco ? ......................

Ora accendiamo il LED interno quando la graffetta è toccata

|  |  |
| --- | --- |
| tp = TouchPad(Pin(4))  SOGLIA = ...  while True:      if tp.read() > SOGLIA:          Pin(10).on()      else:          Pin(10).off()      sleep\_ms(250) | **Thonny/main.py**  Sostituire i tre puntini con il valore scelto sopra |

Ora accendiamo il LED interno e un messaggio quando la graffetta è toccata

|  |  |
| --- | --- |
| tp = TouchPad(Pin(4))  SOGLIA = ...  while True:      if tp.read() > SOGLIA:          Pin(10).on()          display.text('Ciao!', 40, 12, 1)          display.show()      else:          Pin(10).off()          display.text('Arrivederci!', 40, 12, 1)          display.show()      sleep\_ms(250) | Thonny/main.py  Sostituire i tre puntini con il valore scelto sopra |

**Esperimento 3 – Lampeggi LED esterno**

|  |  |
| --- | --- |
| **Circuito** | **Istruzioni di collegamento** |
| Immagine che contiene circuito, elettronica, Ingegneria elettronica, Componente elettrico  Descrizione generata automaticamente**Immagine che contiene schermata, linea, numero, testo  Descrizione generata automaticamente** | Collegare un cavo fra il pin 36 e una resistenza di almeno 1kΩ  Collegare la resistenza all’anodo del diodo  Collegare il catodo del diodo alla fila MENO orizzontale della breadboard  Collegare il GND della board alla fila MENO orizzontale della breadboard |

|  |  |
| --- | --- |
| **Lampeggio sincronizzato**  LED esterno e LED integrato ogni secondo | **Lampeggio alternato**  LED esterno e LED integrato ogni secondo |
| **Thonny / main.py**  led1 = Pin(36, Pin.OUT)  led2 = Pin(10, Pin.OUT)  while True:      led1.on()      led2.on()      sleep(1)      led1.off()      led2.off()      sleep(1) | **Thonny / main.py**  led1 = Pin(36, Pin.OUT)  led2 = Pin(10, Pin.OUT)  while True:      led1.on()      led2.off()      sleep(1)      led1.off()      led2.on()      sleep(1) |

**Esperimento 4 – Programmazione asincrona**

|  |  |
| --- | --- |
| **Circuito** | **Istruzioni di collegamento** |
| Immagine che contiene circuito, elettronica, Ingegneria elettronica, Componente elettrico  Descrizione generata automaticamente**Immagine che contiene schermata, linea, numero, testo  Descrizione generata automaticamente** | Stesso circuito dell’esperimento 3 |
| Lampeggio con periodi diversi (asincrono) | Approccio classico |
| import asyncio  led1 = Pin(36, Pin.OUT)  led2 = Pin(10, Pin.OUT)  async def blink\_1():      global led1      while True:          led1.value(not(led1.value()))          await asyncio.sleep\_ms(2000)  async def blink\_2():      global led2      while True:          led2.value(not(led2.value()))          await asyncio.sleep\_ms(1000)  def main():      t1 = asyncio.create\_task(blink\_1)      t2 = asyncio.create\_task(blink\_2)      asyncio.gather(t1, t2)  (Thonny/main.py) | led1 = Pin(36, Pin.OUT)  led2 = Pin(10, Pin.OUT)  while True:      led1.on()      led2.off()      sleep(1)      led1.on()      led2.on()      sleep(1)      led1.off()      led2.off()      sleep(1)      led1.off()      led2.on()      sleep(1)  (Thonny/main.py) |

Fin’ora sembra un po’ la stessa cosa, ma più complessa.

Aggiungiamo un contatore di secondi sull’LCD

|  |  |
| --- | --- |
| Approccio asincrono | Approccio classico |
| import asyncio  led1 = Pin(36, Pin.OUT)  led2 = Pin(10, Pin.OUT)  contatore = 0  async def blink\_1():      global led1      while True:          led1.value(not(led1.value()))          await asyncio.sleep\_ms(2000)  async def blink\_2():      global led2      while True:          led2.value(not(led2.value()))          await asyncio.sleep\_ms(1000)  async def conta():      global contatore      while True:          contatore += 1          display.text(str(contatore), 40, 12, 1)          display.show()          await asyncio.sleep\_ms(1000)  def main():      t1 = asyncio.create\_task(blink\_1)      t2 = asyncio.create\_task(blink\_2)      t3 = asyncio.create\_task(conta)      asyncio.gather(t1, t2, t3) | contatore = 0  led1 = Pin(36, Pin.OUT)  led2 = Pin(10, Pin.OUT)  while True:      led1.on()      led2.off()      contatore += 1      display.text(str(contatore), 40, 12, 1)      display.show()      sleep(1)      led1.on()      led2.on()      contatore += 1      display.text(str(contatore), 40, 12, 1)      display.show()      sleep(1)      led1.off()      led2.off()      contatore += 1      display.text(str(contatore), 40, 12, 1)      display.show()      sleep(1)      led1.off()      led2.on()      contatore += 1      display.text(str(contatore), 40, 12, 1)      display.show()      sleep(1) |

Se ora dovessi cambiare il lampeggio del primo led a 0,5 s?

Basterebbe cambiare una riga nel primo caso (asincrono), mentre dovrei riscrivere quasi tutto il secondo programma (sincrono).

**Aiuto coordinate dado**

|  |  |
| --- | --- |
| 32  32 | Asse X 🡪  Asse  Y  |  \/ |
| Ellisse | Elisse centrata in posizione (0,0) |

**Esperimento 1 : generare numeri casuali**

|  |  |
| --- | --- |
| Python | Output programma (6 tiri) |
| import random  print(random.randint(1,6)) | ..........................................  ..........................................  ..........................................  ..........................................  ..........................................  .......................................... |

Rappresentare i vostri risultati come un istogramma colorando il numero di volte in cui è uscito ogni risultato

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

**Esperimento 2: disegnare sullo schermo**

|  |  |
| --- | --- |
| Quadrato | display.fill\_rect(0, 0, 32, 32, 1) |
| Cerchio | display.ellipse(16, 16, 4, 4, 0, True)  Centro del cerchio : X = 16, Y = 16  Semiasse orizzontale : 4 pixels  Semiasse verticale : 4 pixels  Colore dell'elisse : 0  Ultimo parametro True per riempire l’elissi |
| Disegnare cerchi da una lista | # Facciamo una lista di coordinate (tuple)  centri = [(16,16),(8,8),(24,24)]  for centro in centri:      display.ellipse(centro[0], centro[1], 4, 4, 0, True) |

Numero scelto per il gruppo : ....................................

|  |  |
| --- | --- |
| Numero | Coordinate dei centri dei puntini |
| 1 | ( ..... , ..... ) |
| 2 | ( ..... , ..... ) ( ..... , ..... ) |
| 3 | ( ..... , ..... ) ( ..... , ..... ) ( ..... , ..... ) |
| 4 | ( ..... , ..... ) ( ..... , ..... ) ( ..... , ..... ) ( ..... , ..... ) |
| 5 | ( ..... , ..... ) ( ..... , ..... ) ( ..... , ..... )  ( ..... , ..... ) ( ..... , ..... ) |
| 6 | ( ..... , ..... ) ( ..... , ..... ) ( ..... , ..... )  ( ..... , ..... ) ( ..... , ..... ) ( ..... , ..... ) |

**Esperimento 3: dato un numero, disegnare il dado corrispondente**

|  |
| --- |
| def disegna\_dado(valore\_tratto):      display.fill(0)      display.fill\_rect(0, 0, 32, 32, 1)      dado = { 1: [(16,16)],               2: [(8,8), (24,24)],               3: [(8,8), (16,16), (24,24)],               4: [(8,8), (8,24), (24,8), (24,24)],               5: [(8,8), (8,24), (24,8), (24,24), (16,16)],               6: [(8,8), (8,24), (24,8), (24,24), (8,16), (24,16)] }        puntini = dado[valore\_tratto]      for punto in puntini:          display.ellipse(punto[0], punto[1], 4, 4, 0, True)      display.show()  # Proviamo con un numero  disegna\_dado(6) |

Disegnare il risultato ottenuto

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

**Esperimento 4 : conclusione dado elettronico**

|  |  |
| --- | --- |
| Immagine che contiene schizzo, testo, strumento, sega  Descrizione generata automaticamente | **Graffetta su Pin4** |
| def disegna\_dado(display, value, pos = 0):      WIDTH = 32      RADIUS = 4      if pos > 0:          xmin = (WIDTH + 2) \* pos      else:          xmin = 0      display.fill\_rect(xmin, 0, WIDTH, WIDTH, 1)      coordinate = { 1: [(16,16)],                     2: [(8,8), (24,24)],                     3: [(8,8), (16,16), (24,24)],                     4: [(8,8), (8,24), (24,8), (24,24)],                     5: [(8,8), (8,24), (24,8), (24,24), (16,16)],                     6: [(8,8), (8,24), (24,8), (24,24), (8,16), (24,16)] }        for center in coordinate[value]:          display.ellipse(xmin + center[0], center[1], RADIUS, RADIUS, 0, True)      display.show()    cpt = 0  led = Pin(10, Pin.OUT)  touch = TouchPad(Pin(4))  NUM\_DADI = 3  while True:      if touch.read() > 10000:          display.fill(0)          led.value(not led.value())          for num\_dado in range(0, NUM\_DADI):              disegna\_dado(display, random.randint(1, 6), num\_dado)      time.sleep\_ms(250)      cpt += 1 | |