Περιεχόμενα

Προετοιμασία

Λογισμικό

Thonny: Για την συγγραφή του κώδικα Python για το Raspberry Pi Pico.

Tinkercad: Για την σχεδίαση των δομικών στοιχείων.

Prusa Slicer: Για την δημιουργία του gcode για τον 3D εκτυπωτή.

Εξοπλισμός

Απαραίτητος εξοπλισμός:

- Σταθμός συγκόλλησης
- 3D εκτυπωτής

Βοηθητικός εξοπλισμός:

- Θερμόκολλα

Υλικά

Quantity	Description
1	Li-ion Battery Charger Protection Module 2S 3A
1	Μετατροπέας DC-DC Step-Up 5-35V 2A
2	Μπαταριοθήκη 1x18650 - με Καλώδια
2	Μπαταρία Λιθίου 18650 3.7V
1	Raspberry Pi Pico
2	Pin Header 1x40 Male 2.54 mm Black
1	Voltage Regulator L7805CV - 5V 1.5A
2	Servo Micro 1.8kg.cm Metal Gears (Feetech FS90MG)
4	Φωτοαντίσταση LDR 5mm
2	Πλακέτα Διάτρητη 91x45mm (σπάει σε 2 45x45 η κάθε μία)
4	Αντίσταση Carbon 1/4W 5% 1Kohm
4	Φωτοβολταϊκή Κυψέλη 1W 125x63mm
7	JST XH Jumper 2 Wire Assembly - 20cm
7	JST XH Conector 2-Pin Male 2.5mm
1	JST XH Jumper 3 Wire Assembly - 20cm
1	JST XH Conector 3-Pin Male 2.5mm

4	Schottky Rectifier - 1A 1N5819
1	PrimaSelect PLA PRO Filament - 1.75mm - 750g spool - Black
2	Αντίσταση Carbon 2W 5% 1ohm
1	MCP3004 - 10bit 4 channel ADC SPI
1	Θερμοσυστελλόμενο 6.4/3.2mm Διάφανο - 1m

Σενάριο

Καθώς η ηλεκτρική ενέργεια αποτελεί ένα απαραίτητο αγαθό στον σημερινό τρόπο ζωής μας, σκεφτήκαμε να υλοποιήσουμε ένα σύστημα παραγωγής πράσινης ενέργειας. Η υλοποίηση βασίζεται στην τεχνολογία των φωτοβολταϊκών κυψελών και το σύστημα θα σχεδιαστεί με τρόπο που να μεγιστοποιεί την παραγωγή ενέργειας από τον ήλιο καθ' όλη τη διάρκεια της μέρας.

Προσέγγιση προβλήματος

Οι φωτοβολταικές κυψέλες παράγουν ηλεκτρική ενέργεια μέσω της ηλιακής ακτινοβολίας και η ισχύς που παράγουν εξαρτάται διάφορους παράγοντες όπως η θέση τους σε σχέση με τον ήλιο, ο καιρός, τα χρόνια λειτουργίας των κυψελών κτλ.

Προφανώς, κάποιους από τους παράγοντες όπως ο καιρός δεν μπορούμε να τους ελέγξουμε. Υπάρχουν όμως κάποιοι παράγοντες τους οποίους μπορούμε να βελτιστοποιήσουμε, για παράδειγμα την κλίση του συστήματός μας ως προς τον ήλιο.

Συνεπώς, η βασική ιδέα για την βελτιστοποίηση της παραγωγής ηλεκτρικής ενέργειας μέσω φωτοβολταϊκών κυψελών είναι η αυτόματη αναπροσαρμογή της κλίσης των κυψελών ως προς τον ήλιο.

Έρευνα

Ο ήλιος αλλάζει θέση διαρκώς σε σχέση με εμάς. Η τροχιά που βλέπουμε να διαγράφει ο ήλιος σε σχέση με τη γη, η οποία μας είναι ιδιαίτερα οικεία, είναι η κίνηση από την ανατολή προς την δύση κατά τη διάρκεια της μέρας. Αυτό που ίσως να μην έχουμε παρατηρήσει είναι η διαφορετική θέση του ήλιου στον ουράνιο θόλο κατά την αλλαγή των εποχών!

Από παρατήρηση τις θέσης του ήλιου κατά τη διάρκεια της μέρας, συμπεραίνουμε ότι το σύστημα που θα υλοποιήσουμε θα χρειαστεί δύο βαθμούς ελευθερίας. Θα χρειαστεί δηλαδή να μπορεί να βελτιστοποιεί τη θέση των κυψελών ως προς τον ήλιο κατά τη διάρκεια της μέρας ακολουθώντας την πορεία του από ανατολή μέχρι δύση, αλλά θα πρέπει να κάνει και διορθώσεις για την αλλαγή του «ύψους» του ήλιου. Οι διορθώσεις «ύψους» σε σταθερά πάνελ γίνονται μόνο μερικές φορές τον χρόνο, αλλά καθώς το δικό μας σύστημα θα λειτουργεί αυτόματα, οι διορθώσεις θα γίνονται καθ' όλη τη διάρκεια της μέρας, στοχεύοντας έτσι τον ήλιο ανά πάσα στιγμή.

Το τελικό συμπέρασμα για την υλοποίηση είναι ότι το σύστημα θα πρέπει να μπορεί να κινεί τις κυψέλες δεξιά/αριστερά και πάνω/κάτω.

Σχεδιασμός συστήματος

Για το σύστημά μας θα επιλέξαμε 4 Φωτοβολταϊκές κυψέλες 1W 125x63mm οι οποίες δίνουν περίπου 6V.

Για το μικροελεγκτή θα χρησιμοποιήσουμε το raspberry pi pico, το οποίο λειτουργεί με τάση 5V για τροφοδοσία και τάση 3.3V στους ακροδέκτες.

Τα θέματα που δημιουργούνται είναι τα εξής:

- Οι μπαταρίες πρέπει να είναι επαναφορτιζόμενες
- Μία μπαταρία δεν δίνει αρκετή τάση για να τροφοδοτεί με 5V τα ηλεκτρονικά μας
- Οι φωτοβολταϊκές κυψέλες δεν δίνουν αρκετή τάση για να φορτίσουν δύο μπαταρίες αν αυτές είναι σε σειρά
- Οι μπαταρίες λιθίου χρειάζονται κύκλωμα προστασίας για την ασφαλή φόρτιση και εκφόρτιση

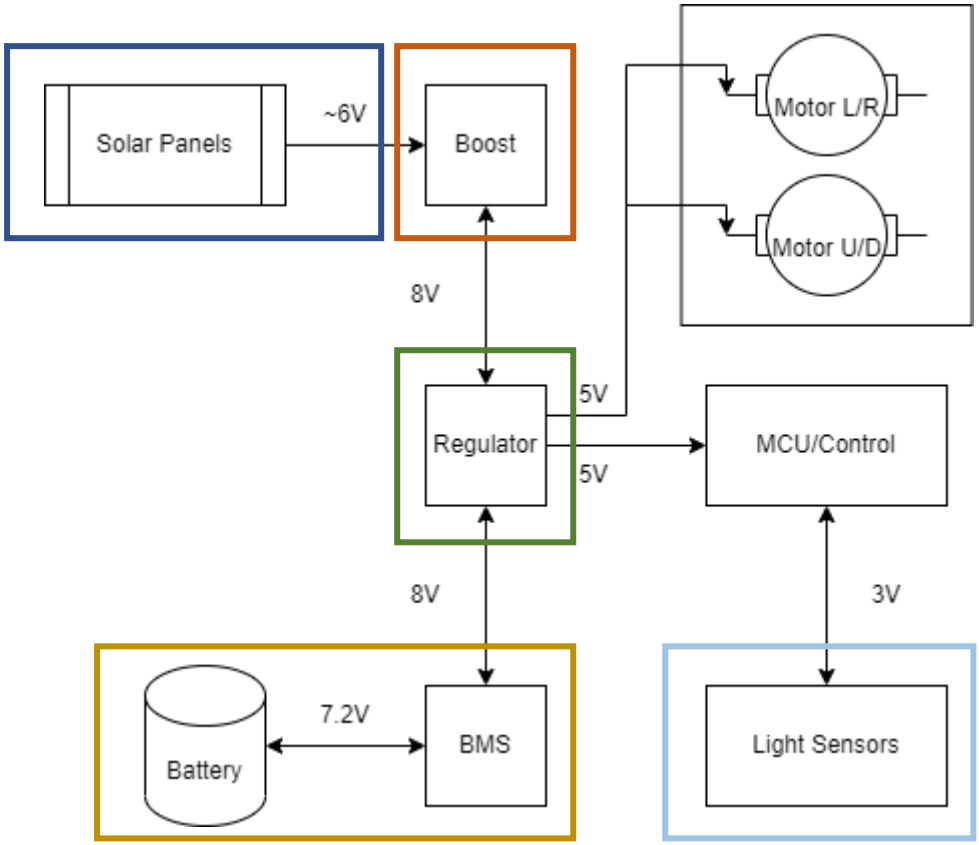
Επιλέγουμε τα πάνελ να τα συνδέσουμε παράλληλα ώστε να κρατήσουμε την τάση σχετικά χαμηλά και να μεγαλώσουμε το ρεύμα (δύο πάνελ σε σειρά δίνουν διπλάσια τάση, δύο πάνελ παράλληλα δίνουν διπλάσιο ρεύμα σε ιδανικές συνθήκες).

Στην περίπτωση που τα πάνελ δεν παράγουν αρκετή ενέργεια, θέλουμε το σύστημά μας να συνεχίσει να λειτουργεί, συνεπώς θα χρειαστούμε τουλάχιστον δύο μπαταρίες σε σειρά για να δημιουργήσουμε τάση μεγαλύτερη των 5V.

Για να μπορούμε να φορτίσουμε δύο μπαταρίες σε σειρά χρειαζόμαστε μία τάση μεταξύ 8V και 8.4V. Αυτό σημαίνει ότι θα χρειαστεί να βρούμε ένα κατάλληλο σύστημα το οποίο δέχεται μία τάση μικρότερη της ιδανικής (η τάση από τα πάνελ) και την μετατρέπει στην μεγαλύτερη ιδανική (η τάση που φορτίζουν οι μπαταρίες). Τέτοια συστήματα ονομάζονται Boost Converters. Η έξοδος του συστήματος αυτού θέλουμε να είναι μεταξύ 8V και 8.4V.

Το τελευταίο θέμα που χρειάζεται να λύσουμε είναι η τάση τροφοδοσίας για τον ελεγκτή μας, η οποία πρέπει να είναι 5V. Για τον σκοπό αυτό θα βρούμε ένα σύστημα το οποίο παίρνει μία τάση μεγαλύτερη από την απαιτούμενη (για εμάς θα δέχεται γύρω στα 8V) και θα την υποβιβάζει σε σταθερά 5V. Αυτά τα συστήματα λέγονται voltage regulators.

Στο σύστημά μας θα χρησιμοποιήσουμε δύο κινητήρες servo για τους δύο βαθμούς ελευθερίας (καθώς είναι αρκετά εύχρηστοι και οικονομικοί) και 4 LDR (light dependent resistor) για την αναγνώριση της θέσης του ήλιου.



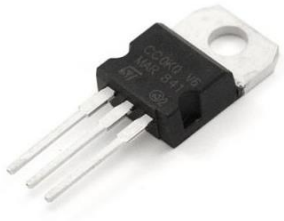
Επιλογή βασικών στοιχείων

Φωτοβολταϊκή Κυψέλη 1W 125x63mm

Μετατροπέας DC-DC Step-Up 5-35V 2A (boost converter)

Li-ion Battery Charger Protection Module 2S 3A

Voltage Regulator L7805CV - 5V 1.5A

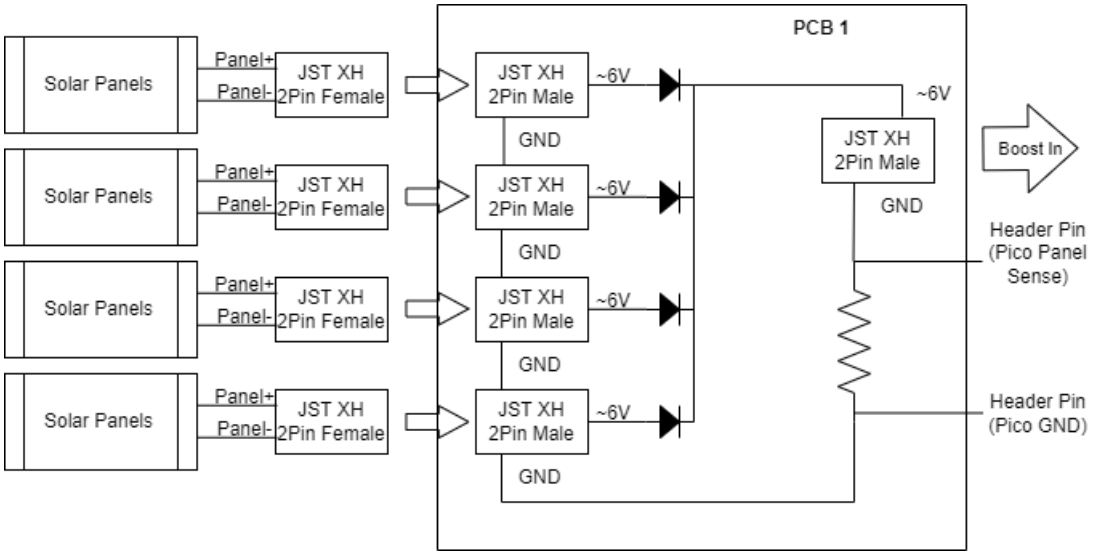


Servo Micro 1.8kg.cm Metal Gears (Feetech FS90MG)

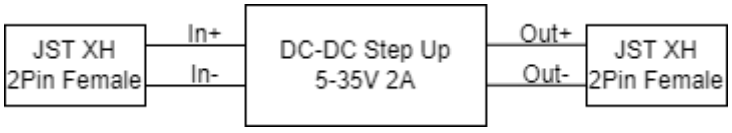
Μπαταρία Λιθίου 18650 3.6V 3200mAh - LG INR18650-MH1

Σχεδιασμός υλικού

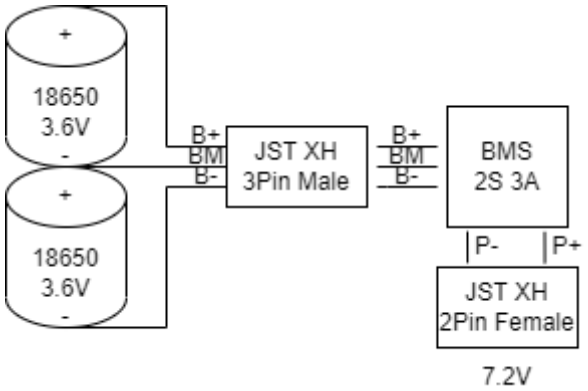
Πάνελ



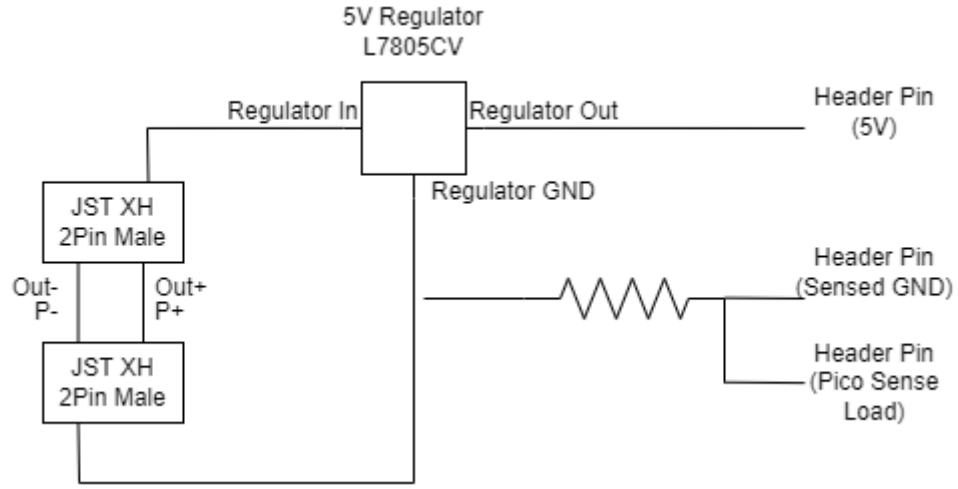
Ανύψωση τάσης (Boost DC/DC)



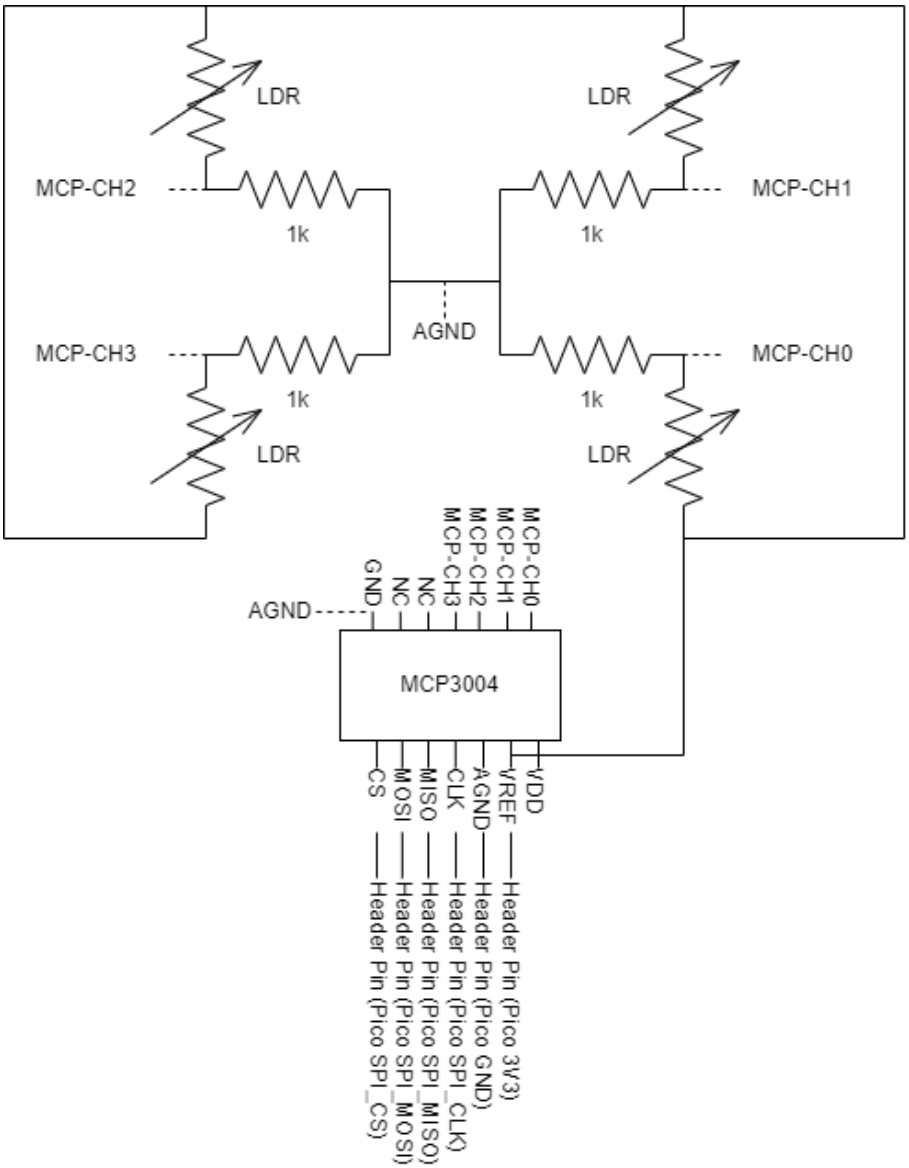
Μπαταρίες



Δημιουργία σταθερής τάσης 5V (Voltage regulator)



Αισθητήρας θέσης Ήλιου

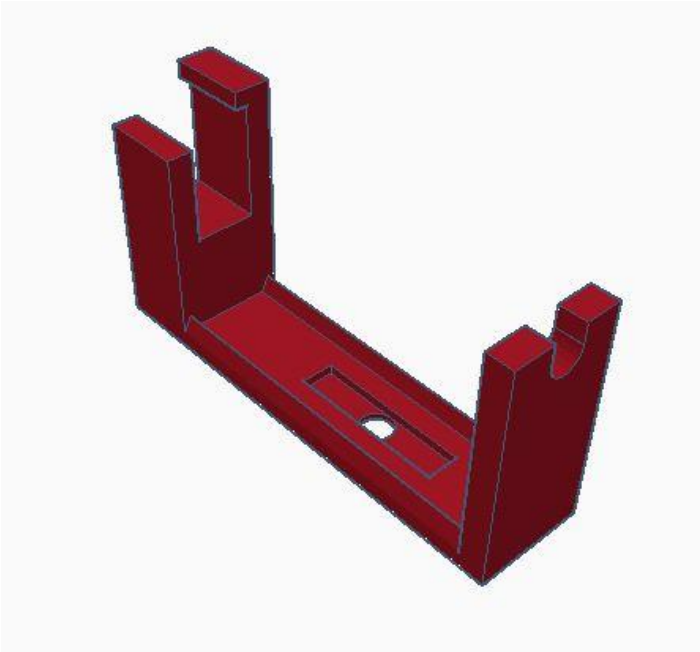


Σχεδιασμός δομικών στοιχείων

Βάση

Η βάση σχεδιάζεται ώστε να παρέχει επαρκή στατικότητα στο σύστημά μας. Στο κέντρο της βάσης δημιουργούμε οπή στις διαστάσεις του κινητήρα servo.

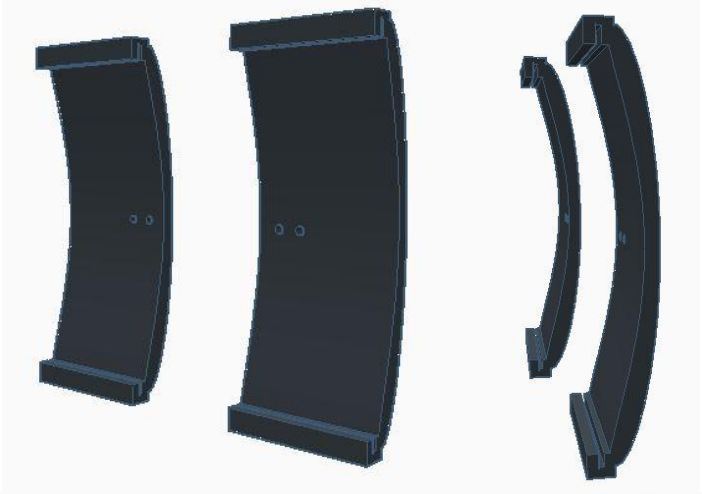
Σύνδεσμος servo βάσης

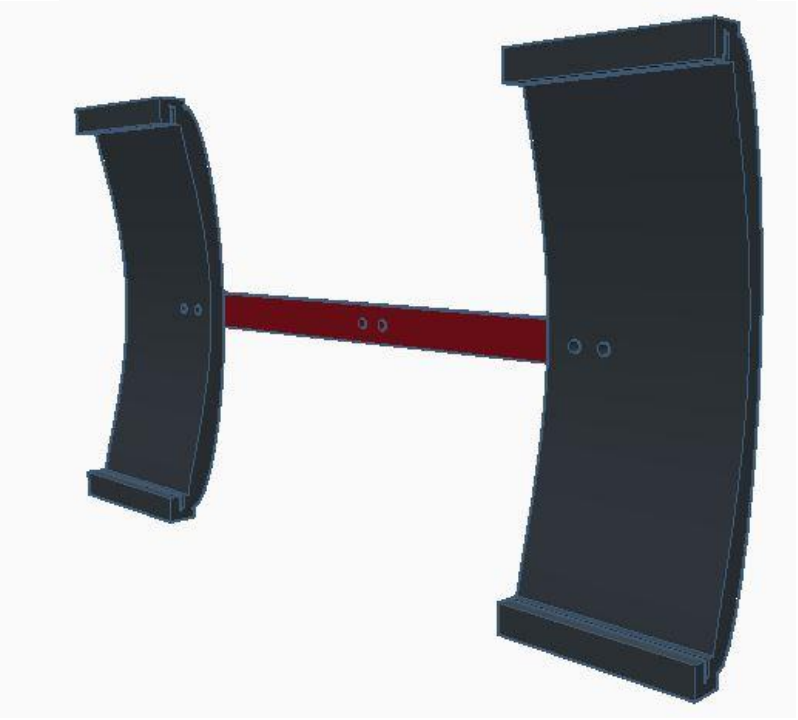


Ο σύνδεσμος βάσης τοποθετείται πάνω στον κινητήρα βάσης και επιτρέπει να περιστροφή των πάνελ δεξιά και αριστερά. Στη μία μεριά υπάρχει εσοχή για τον δεύτερο κινητήρα servo

και από την άλλη έχει σχεδιαστεί εγκοπή για να μπορεί να υπάρχει υποστήριξη των πάνελ και να στέκονται ίσια.

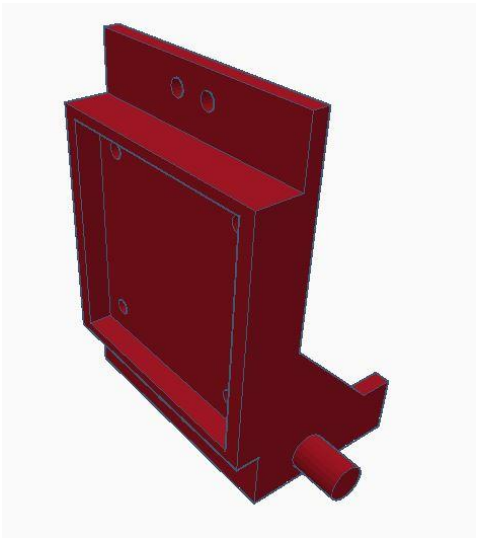
Σύνδεσμος πάνελ

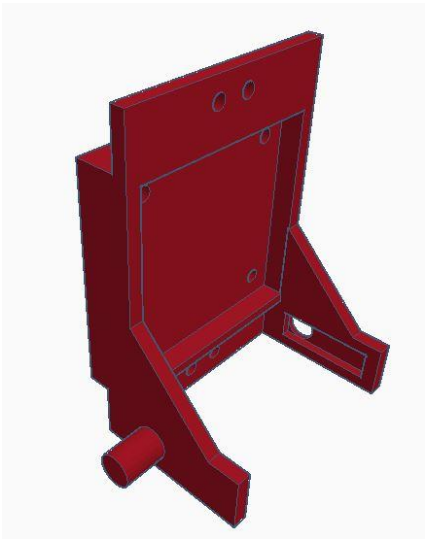


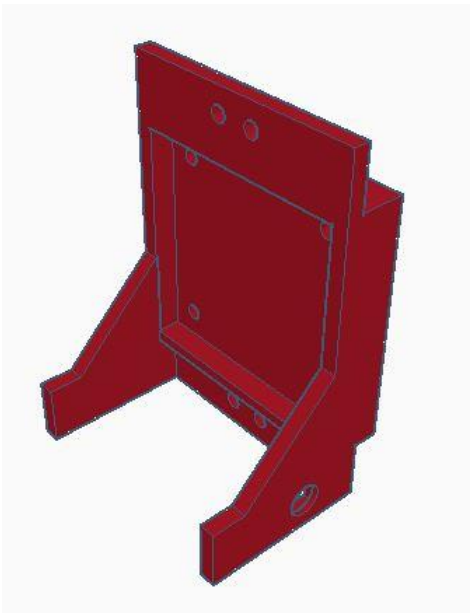


Τα πάνελ εφαρμόζουν συρταρωτά, δύο σε κάθε βάση πάνελ, και οι δύο αυτές βάσης βιδώνονται πάνω στον άξονα στήριξης.

Σύνδεσμος servo κλίσης







Στο κεντρικό εξάρτημα θα βιδωθεί ο άξονας με τις βάσεις των πάνελ (και τα ίδια τα πάνελ) στις δύο κάτω οπές. Στο αρχικό σχέδιο βιδώνονταν στις πάνω, αλλά το κέντρο βάρους του συστήματος ήταν μακριά από τον άξονα περιστροφής, με αποτέλεσμα να χρειάζεται περισσότερη ενέργεια για να μετακινείται το σύστημα.

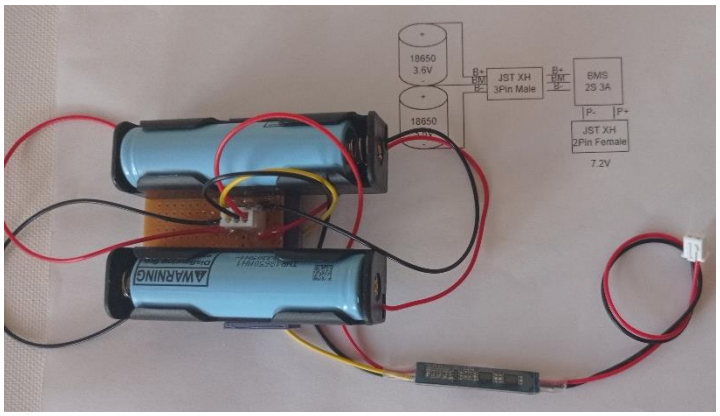
Ο κύλινδρος στο κάτω μέρος της μίας μεριάς παρέχει υποστήριξη του βάρους των πάνελ. Στην άλλη μεριά βλέπουμε την εσοχή για τον άξονα του κινητήρα που επιτρέπει να αλλαγή της κλίσης του συστήματος.

Στην μπροστά τετράγωνη εσοχή, τοποθετείται η πλακέτα με τους αισθητήρες φωτός και στην πίσω η πλακέτα στην οποία συνδέονται τα πάνελ.

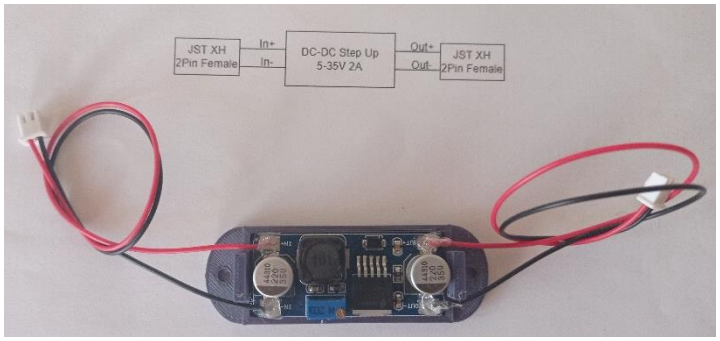
Υλοποίηση υλικού

Το υλικό μπορεί να υλοποιηθεί σε διάτρητες πλακέτες, το οποίο αν και είναι πιο δύσκολο στην υλοποίηση προτείνεται λόγω της σθεναρότητας που θα έχει έτσι το σύστημα. Διαφορετικά, όλες οι συνδεσμολογίες γίνεται σε breadboard, το οποίο είναι πιο εύκολο να υλοποιηθεί, αλλά συνήθως πολλά καλώδια χρειάζονται επανασύνδεση λόγω της έλλειψης κολλήσεων.

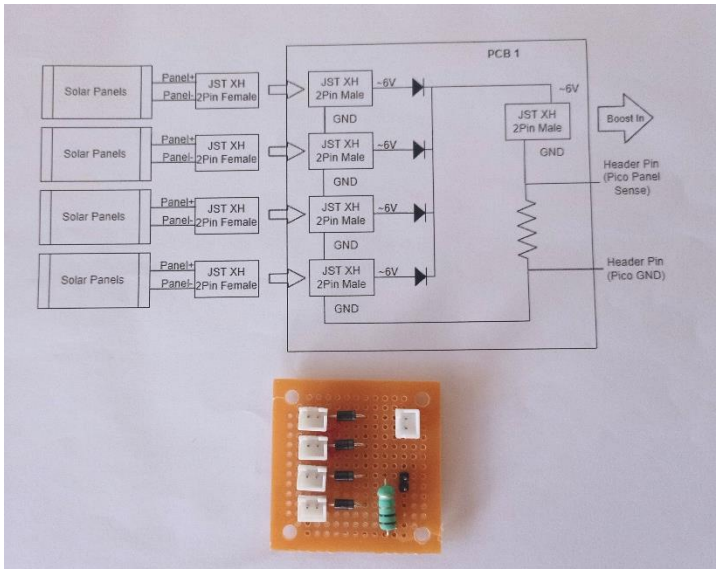
Μπαταρίες και BMS



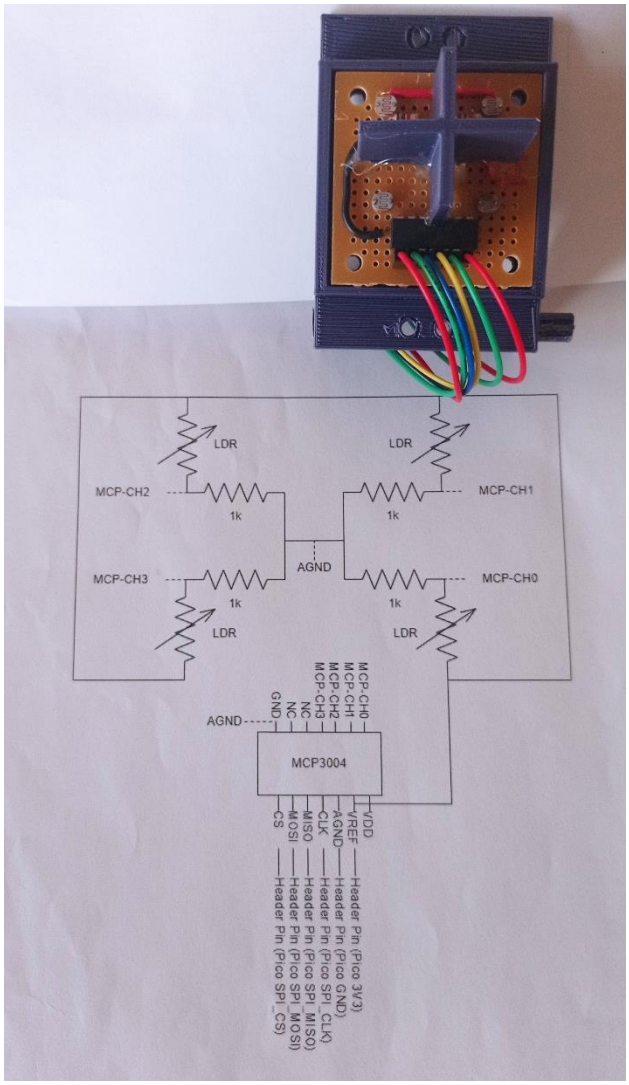
Ανύψωση τάσης (Boost DC/DC)



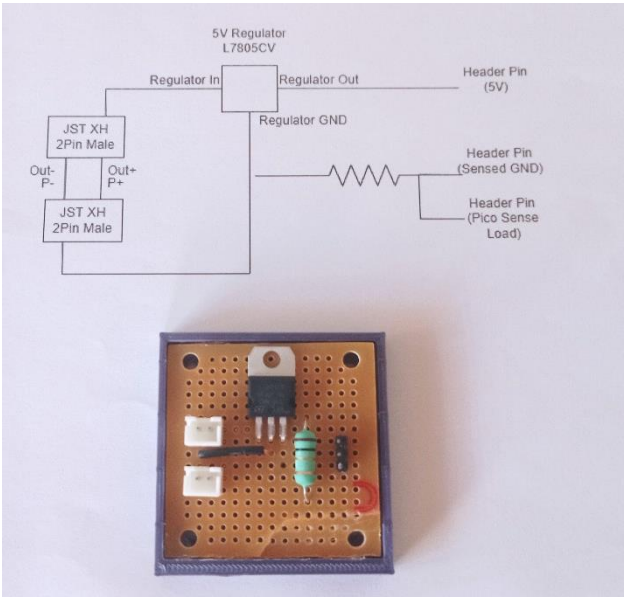
Συλλέκτης Πάνελ



Αισθητήρας θέσης Ήλιου



Δημιουργία σταθερής τάσης 5V (Voltage regulator)



Σχεδιασμός λογισμικού

Σημείωση: Για τη λειτουργία του προγράμματος χρησιμοποιείται η βιβλιοθήκη mcp3004. Για να έχουμε πρόσβαση σε αυτή, θα πρέπει το αρχείο mcp3004.py να περαστεί στη μνήμη flash του Raspberry Pi Pico.

Φόρτωση βιβλιοθηκών.

```
from time import *
from machine import *
from mcp3004 import MCP3004
```

Αρχικοποίηση ακροδεκτών για το SPI.

```
""" SPI Parameters """
SPI_CLK = Pin(2)
SPI_MOSI = Pin(3)
SPI_MISO = Pin(4)
SPI_CS = Pin(5, Pin.OUT)
```

Αρχικοποίηση SPI και βιβλιοθήκης του ολοκληρωμένου MCP3004.

```
spi = SPI(0, sck=SPI_CLK, mosi=SPI_MOSI, miso=SPI_MISO, baudrate=100000)
cs = SPI_CS

mcp = MCP3004(spi, cs)
```

Αρχικοποίηση ακροδεκτών για αναλογική ανάγνωση.

```
""" ADC Parameters """
SPANEL_PIN = Pin(26, Pin.IN) # Sense PANEL PIN
SPOUT_PIN = Pin(27, Pin.IN)  # Not Used! Sense Power OUT PIN

sPanel = ADC(SPANEL_PIN)
sPout = ADC(SPOUT_PIN)
```

Αρχικοποίηση ακροδεκτών σε PWM 50Hz για έλεγχο της θέσης των κινητήρων servo.

```
""" Control Parameters """
SERVO_LR_PIN = Pin(15, Pin.OUT)
SERVO_UD_PIN = Pin(14, Pin.OUT)

servoLR = PWM(SERVO_LR_PIN)
servoUD = PWM(SERVO_UD_PIN)
servoLR.freq(50) # 50Hz
servoUD.freq(50) # 50Hz
```

Αρχικοποίηση λοιπών παραμέτρων συστήματος.

```
""" System Parameters """
onboardLED = Pin(25, Pin.OUT)
```

```
SYS_ADC_ACCURACY = 0xFFFF  # system ADC is 16 bit
MCP_ADC_ACCURACY = 0xFFF   # mcp ADC is 12 bit
```

Πλήθος αισθητήρων φωτός.

```
NUM_LDR = 4 # number of Light Dependent Resistors in our system
```

Το σύστημά μας λειτουργεί στα 3.3V. Η τάση αναφοράς χρειάζεται για να μετατρέψουμε τιμή βήματος του ADC σε τιμή τάσης σε mV.

```
SYS_mV_REF = 3300 # system votlage reference in mV
```

Η τιμή της αντίστασης μέτρησης ρεύματος χρειάζεται για να μετατρέψουμε την αναλογική τάση σε τιμή ρεύματος σε mA.

```
SENSE_RES_VALUE = 1  # value of sensing resistors
```

Ο χρόνος που το σύστημα θα πέσει σε καταστολή μέχρι να χρειαστεί να πάρει νέα μέτρηση.

```
SLEEP_TIME = 2000 # changes if isDemo is not 0
```

Η διαφορά που χρειάζεται να έχουν οι αισθητήρες μεταξύ τους ώστε το σύστημα να διορθώσει την θέση του ως προς τον ήλιο.

```
SENSITIVITY = 3
```

Η μοίρες που μετακινεί τον αντίστοιχο κινητήρα για κάθε διόρθωση.

```
SERVO_STEP = 1
```

Αντιστοίχιση καναλιών MCP σε θέση αισθητήρα.

```
# sensor-mcp channel mapping
LDR_UR = 1 # up right ldr when looking from the front
LDR_UL = 2 # up left ldr when looking from the front
LDR_DR = 0 # down right ldr when looking from the front
LDR_DL = 3 # down left ldr when looking from the front
```

Ακραίες τιμές κινητήρων για δεξιά-αριστερά, πάνω-κάτω.

```
LR_MIN = 30
LR_MAX = 150
UD_MIN = 90
UD_MAX = 150


""" ======================
=== System and functions ===
======================== """
```

Υπολογίζει τη σχέση μεταξύ των διαφορετικών αισθητήρων για δεδομένη ένταση φωτός. Για να γίνει σωστά ο υπολογισμός θα πρέπει, όταν κληθεί αυτή η συνάρτηση, οι αισθητήρες να είναι εκτεθειμένοι στην ίδια ένταση φωτός.

```
def CalcSensorNormalization():
  acc_ldr = [0, 0, 0, 0]

  for i in range(10):
    sensorValues = ReadLightSensors_raw()
```

```
            for j in range(NUM_LDR):
                acc_ldr[j] += sensorValues[j]
            sleep_ms(100)

        for i in range(NUM_LDR):
            print(acc_ldr[i], end=' ')
        print()

        normValues = [1., 1., 1., 1.]
        for i in range(NUM_LDR):
            normValues[i] = round(sensorValues[0] / sensorValues[i], 2)
            print(normValues[i], end=" ")
        print()
```

Διαβάζει την αναλογική τάση από τον ADC που δίνεται σαν όρισμα και τη μετατρέπει σε τιμή ρεύματος σε mA.

```
    def ReadCurrent(sourceADC):
        rawValue = sourceADC.read_u16()
        voltage = rawValue * SYS_mV_REF / SYS_ADC_ACCURACY
        current = voltage / SENSE_RES_VALUE # I = V/R (R=1 in our case)

        return round(current) # return current in mA


    """ ================================
    === Energy production monitoring ===
    ================================== """
```

Υπολογίζει την παραγόμενη ισχύ βάση του μετρούμενου παραγόμενου ρεύματος.

```
    def UpdateEnergyValues():
        global panelCurrent, panelWatt

        panelCurrent = (panelCurrent + ReadCurrent(sPanel)) >> 1 # division by shifting
        panelWatt = pow(panelCurrent, 2) * SENSE_RES_VALUE / 1000 # divide by 1000
    because of mA ^ 2
        panelWatt = round(panelWatt)
```

Υπολογίζει την συνολική ενέργεια που έχει παράξει το σύστημα όσο λειτουργεί.

```
    def CalcPowerProduction():
        global productionWatt, productionTime, productionWSeconds

        productionWatt += panelWatt # in mW
        productionTime += SLEEP_TIME # in ms
        productionWSeconds = round(productionWatt * (productionTime/1000) / 3600,
    2) # power production in mWHours


    """ ================
```

```
    === Read sensors ===
    ==================== """
```
Διαβάζει τις τιμές των αισθητήρων από το MCP.
```
    def ReadLightSensors_raw():
      data = []

      for i in range(NUM_LDR):
        data.append(mcp.read(i))

      return data
```
Διαβάζει τις τιμές των αισθητήρων από το MCP και τις κανονικοποιεί με βάση τη λίστα κανονικοποίησης.
```
    def ReadLightSensors():
      global normValues

      data = []

      for i in range(NUM_LDR):
        data.append(round(mcp.read(i) * normValues[i] * 100 / MCP_ADC_ACCURACY,
    1))

      return data

    """ ============================
    === Correct system position ===
    ============================== """
```
Περιορίζει την τιμή value στο όριο low-high.
```
    def ConstraintValues(value, low, high):
      if value > high:
        return high
      elif value < low:
        return low
      else:
        return value
```
Ελέγχει τον κινητήρα servo για να πάει σε συγκεκριμένη γωνία.
```
    def ServoDegrees(myServo, angle):
      if angle > 180:
        angle = 180
      elif angle < 0:
        angle = 0

      duty = (angle / 180) * 6881.175 + 1310.7
      myServo.duty_u16(int(duty))
```
Ελέγχει και τους δύο κινητήρες.

```
def SetServos(angleUD, angleLR):
  ServoDegrees(servoUD, angleUD)
  ServoDegrees(servoLR, angleLR)
```

Επιστρέφει την τιμή διόρθωσης του συστήματος για το πάνω-κάτω. Η τιμή αυτή εξαρτάται από τις τιμές των αισθητήρων και την ευαισθησία του συστήματος.

```
def CalcServoUD(ldrUR, ldrUL, ldrDR, ldrDL):
  if ldrUR > ldrDR + SENSITIVITY or ldrUL > ldrDL + SENSITIVITY:
    return -SERVO_STEP
  elif ldrUR < ldrDR - SENSITIVITY or ldrUL < ldrDL - SENSITIVITY:
    return SERVO_STEP
  else:
    return 0
```

Επιστρέφει την τιμή διόρθωσης του συστήματος για το δεξιά-αριστερά. Η τιμή αυτή εξαρτάται από τις τιμές των αισθητήρων και την ευαισθησία του συστήματος.

```
def CalcServoLR(ldrUR, ldrUL, ldrDR, ldrDL):
  if ldrUR > ldrUL + SENSITIVITY or ldrDR > ldrDL + SENSITIVITY:
    return SERVO_STEP
  elif ldrUR < ldrUL - SENSITIVITY or ldrDR < ldrDL - SENSITIVITY:
    return -SERVO_STEP
  else:
    return 0


""" ===========
=== Display ===
=============== """
```

Εκτυπώνει στην οθόνη μας χρήσιμες τιμές για παρακολούθηση.

```
def PrintValues():
  print("\nCurrent //", end='\t')
  print("Panel: " + str(panelCurrent) + " mA", end='\n')

  print("Power //", end='\t')
  print("Panel: " + str(panelWatt) + " mW", end='\n')

  for i in range(len(sensorValues)):
    print("LDR" + str(i) + ": " + str(sensorValues[i]), end='\t')
  print()

  print("UD: " + str(angleUD) + "\tLR: " + str(angleLR))


""" ============================
=== Delay and low power mode ===
================================ """
```

Ανάβει για ελάχιστο χρόνο το led που βρίσκεται στο Pico, ώστε να ξέρουμε ότι έγινε λήψη νέας μέτρησης και μετά μπαίνει σε κατάσταση χαμηλής κατανάλωσης μέχρι να περάσει ο υπόλοιπος χρόνος αναμονής.

```
def BlinkAndWait():
  onboardLED.on()
  sleep_ms(50) # wait for pulses to be sent to the servos

  onboardLED.off()
  lightsleep(SLEEP_TIME - 50) # enter low power state and keeping RAM


""" ==========
  Main
  ========== """
ldrUR = 0
ldrUL = 0
ldrDR = 0
ldrDL = 0

angleLR = 90 # set in the middle
angleUD = 135 # set at 45 degrees from being parallel from the ground

panelCurrent = 0
productionWatt = 0
productionTime = 0
```

Εάν το isDemo είναι 1, τότε το σύστημα τρέχει με τις τιμές παρουσίασης οι οποίες έχουν οριστεί με κατάλληλο τρόπο ώστε το σύστημα να λειτουργεί σε χαμηλότερες τιμές φωτεινότητας (εσωτερικός χώρος) και να περιμένει λιγότερο μεταξύ των μετρήσεων.

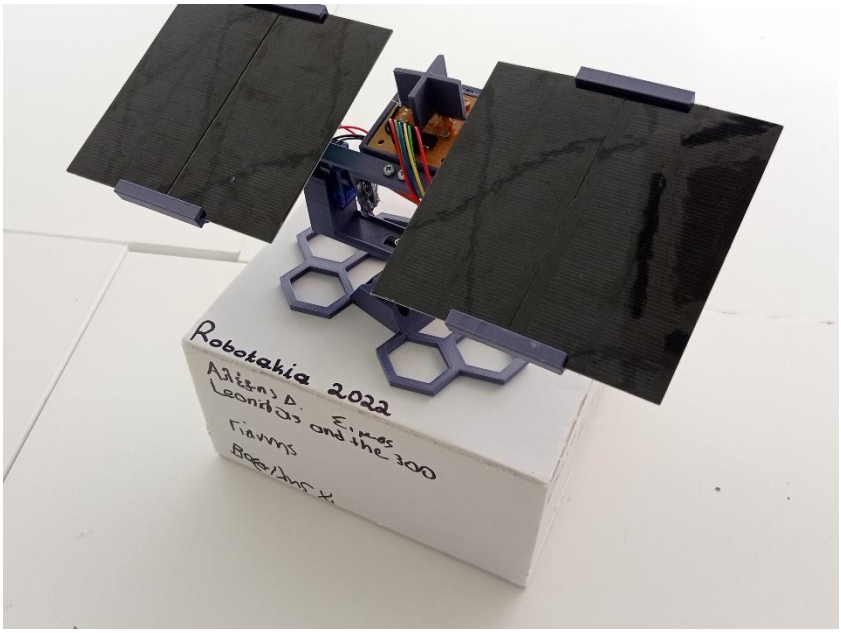
```
# === Select "Is Demo" mode and values ===
isDemo = 1 # set to 0 for normal mode, set to 1 for demo mode
if not isDemo:
  normValues = [1.0, 1.05, 1.01, 1.01] # measured on direct sunlight
  sensorThreshold = 17
else:
  normValues = [1.0, 1.54, 1.1, 1.1] # measured indoors
  sensorThreshold = 7
  SLEEP_TIME = 500


# === System init ===
SetServos(angleUD, angleLR)
```

Βγάζουμε το σχόλιο όταν θέλουμε να βλέπουμε συνέχεια τις τιμές κανονικοποίησης που υπολογίζει το σύστημά μας. Το σύστημα δεν βρίσκεται σε κανονική λειτουργία αν βγάλω τα σχόλια από εδώ.

```
# while True:  # Loop normalization calculation for debugging
#   CalcSensorNormalization()  # Loop normalization calculation for debugging
```

```
CalcSensorNormalization()

while True:
# === Energy production monitoring ===
  UpdateEnergyValues()
  CalcPowerProduction()

# === Read sensors ===
```
Διαβάζουμε τις νέες τιμές αισθητήρων.
```
  sensorValues = ReadLightSensors()
```
Φιλτράρουμε λίγο τις τιμές από θόρυβο βάζοντας μία μικρή αδράνεια στην αλλαγή τους.
```
  ldrUR = (ldrUR + sensorValues[LDR_UR]) / 2
  ldrUL = (ldrUL + sensorValues[LDR_UL]) / 2
  ldrDR = (ldrDR + sensorValues[LDR_DR]) / 2
  ldrDL = (ldrDL + sensorValues[LDR_DL]) / 2
```
Αν οι τιμές αισθητήρων είναι αρκετά μεγάλες ώστε να θεωρούνται αποδεκτές, τότε υπολογίζονται οι διορθώσεις για τους δύο άξονες.
```
# === Correct system position ===
  if max(sensorValues) > sensorThreshold:
    angleUD += CalcServoUD(ldrUR, ldrUL, ldrDR, ldrDL)
    angleUD = ConstraintValues(angleUD, UD_MIN, UD_MAX)

    angleLR += CalcServoLR(ldrUR, ldrUL, ldrDR, ldrDL)
    angleLR = ConstraintValues(angleLR, LR_MIN, LR_MAX)

  SetServos(angleUD, angleLR)

# === Display and save ===
  PrintValues()

# === Delay and low power mode ===
  BlinkAndWait()
```

Υλοποίηση τελικού συστήματος



Τοποθετούμε την βάση σε σταθερό σημείο είτε με κάποια κόλλα, είτε βιδώνοντάς την στις οπές που έχει πάνω σε κάποια μακέτα, κομμάτι ξύλο κτλ.

Τρέχουμε την εντολή SetServos() για τις αρχικές γωνίες σε μία επανάλληψη.

```
while True:
  SetServos(angleUD, angleLR)
```

Έτσι είμαστε βέβαιοι ότι έχουμε αρχικοποιήσει τους κινητήρες προτού τους κολλήσουμε ή βιδώσουμε πάνω στο σύστημα.

Αφού έχουμε αρχικοποιήσει τους κινητήρες στις επιθυμητές αρχικές θέσεις, κολλάμε ή βιδώνουμε τα αξεσουάρ των κινητήρων στις αντίστοιχες εγκοπές του συνδέσμου servo κλίσης και συνδέσμου servo βάσης.

Στο επόμενο βήμα προετοιμάζουμε τα πάνελ.

Βιδώνουμε πρώτα τον άξονα των πάνελ στις βάσεις τους και προσθέτουμε τα πάνελ. Στη συνέχεια το βιδώνουμε στη βάση των αισθητήρων ή οποία μετά τοποθετείται μαζί με τον άξονα του servo στο σύστημα της βάσης που ετοιμάσαμε προηγουμένως.

Τέλος πραγματοποιούμε όλες τις συνδέσεις καλωδίων με προσοχή για να αποφύγουμε ζημιές και λάθη στη λειτουργία του συστήματος.

Συνδέσεις

Module	Module Pin Name	Pico Pin Name	Other Connection	Description
Light Sensor PCB	Pico 3v3	3V3 (OUT)		Power
	Pico GND	GND		Power
	Pico SPI_CLK	GP2		SPI 0
	Pico SPI_MISO	GP4		SPI 0
	Pico SPI_MOSI	GP3		SPI 0
	Pico SPI_CS	GP5		SPI 0
Regulator PCB	Out +		Out + (DC-DC)	Power
	Out -		Out - (DC-DC)	Power
	P +		P + (BMS)	Power
	P -		P - (BMS)	Power
	5V	VBUS		Power
	Sensed GND	No Connect	*User Ground*	
	Pico Sense Load	GP27		ADC
BMS	P +		P + (Regulator PCB)	Power
	P -		P - (Regulator PCB)	Power
	B +		LiPo 7.2V	Power
	BM		LiPo 3.6V	
	B -		LiPo GND	Power
DC-DC Step Up	In +		~6V (Panel)	Power
	In -		GND (Panel)	Power
	Out +		Out + (Regulator PCB)	Power
	Out -		Out - (Regulator PCB)	Power
Panel	~6V		In + (DC-DC)	Power
	GND		In - (DC-DC)	Power
	Pico Panel Sense	GP26		ADC
	Pico GND	GND		Power
Servo Up-Down		GP14		PWM
Servo Left-Right		GP15		PWM