בס"ד

מערכות אלקטרוניות משובצות מחשב

מיני פרויקט – Clock&Temperture

סמסטר ב תשפ"ב, המרכז האקדמי לב

מרצה: דוקטור גולובצ'יוב יוסף יצחק

מגישים: אביאל בירדואקר (207993601) ואופק שרעבי (207206012)

תוכן עניינים

מבוא 2

תיאור המערכת 3

סקיצת המעגל 3

הסברים לשירטוט 4

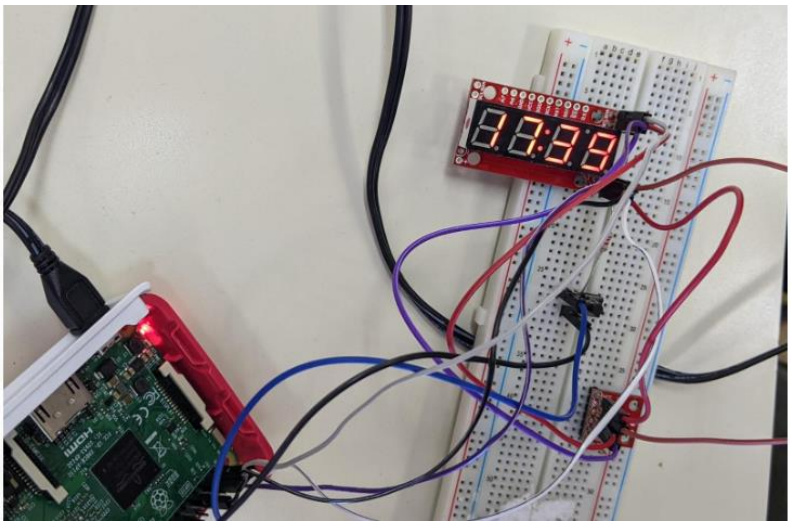
תוכנה 5

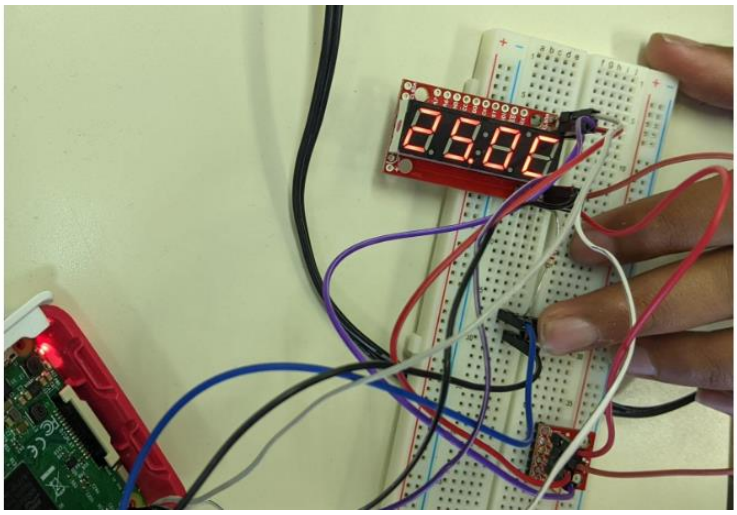
סיכום 8

מבוא

בהתבסס על הרכיבים שהוצעו לנו, בחרנו לבנות שעון בעל פונקציה של מדידת טמפרטורה.
חשבנו על כך שלא נפגשנו אף פעם בשעון יד שיכול לתת את השירות הזה.

במיוחד לאור התקופה האחרונה של מחלת הקורונה , שפקדה אותנו במשך שנתיים ואף יותר, חלק מהזיהוי האם אדם מסוים הוא מאומת או לא הייתה בדיקת חום גופו. כמה נוח היה אם היה ניתן לבדוק את הטמפרטורה בנקל, וזאת על ידי שימוש בשעון שנמצא בהישג יד על יד הנבדק.



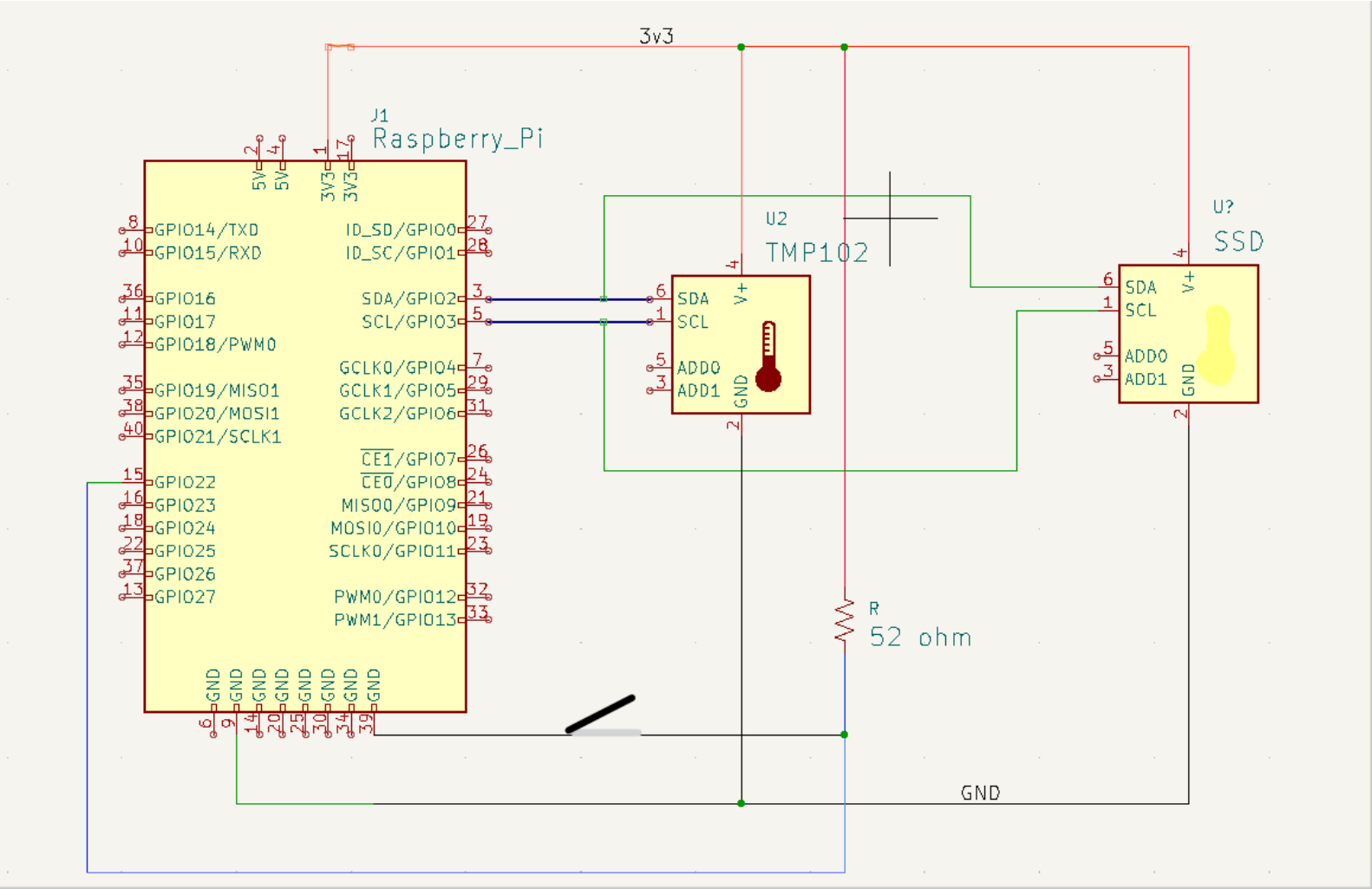


תיאור המערכת

רשימת הרכיבים שבהם השתמשנו:

- Raspberry Pi 3
- מטריצת מעגל אלקטרוני
- כפתור
- Seven Segment Display
- מד טמפרטורה מסוג TMP102
- נגדים
- Jumpers

סקיצת המעגל



הסברים לשירטוט

ראשית, בחרנו לתקשר בין הRP (rasbery pi), הSSD (seven segment display) והTMP על ידי תקשורת מסוג I2C.

לתקשורת מסוג זה יש שימוש ב2 סיגנלים. האחד הוא הDA שזה סיגנל הכתובת והשני הוא הCLK שזה סיגנל השעון. בRP הסיגנלים הללו, נקראים SDA וSCL (בboard הם פינים 3 ו5 בהתאמה).

כיוון שיש רק סוג אחד של ביטים כאלה בRP (השניים הם VC ולא ARM) אנחנו צריכים לחבר בBUS אחד את הSSD (הכתובת שלו היא 0X77) והמד טמפ׳ (הכתובת שלו היא 0X48) כדי שנוכל להשתמש בתקשורת הזה לשניהם יחד.

אנחנו נחבר כפתור, יחד עם נגד PULL-UP, ישירות לRP דרך פין מס׳ 15 בBOARD.

הRP ייתן את הפקודות המתאימות כדי שבSSD נראה את השעה המדויקת (בשעות ודקות). וברגע שנלחץ על הכפתור, הRP ייתן פקודה להציג בSSD את הטמפ׳ המורגשת לחיישן, וזאת למשך 5 שניות, ולאחר מכן יחזור לשעה המדויקת של השעון.

כמובן, אספקת מתח של 3.3 וולט דרך פין 1, ואדמה דרך פין 9.

תוכנה

```python
import time
import smbus
import RPi.GPIO as GPIO

##############################################################################
# Temperture Functions

# Calculate the 2's complement of a number
def twos_comp(val, bits):
    if (val & (1 << (bits - 1))) != 0:
        val = val - (1 << bits)
    return val

# Read temperature registers and calculate Celsius
def read_temp():

    # Read temperature registers
    val = bus.read_i2c_block_data(TMP_ADDRESS, reg_temp, 2)
    # NOTE: val[0] = MSB byte 1, val [1] = LSB byte 2
    #print ("!shifted val[0] = ", bin(val[0]), "val[1] = ", bin(val[1]))

    temp_c = (val[0] << 4) | (val[1] >> 4)
    #print (" shifted val[0] = ", bin(val[0] << 4), "val[1] = ", bin(val[1] >> 4))
    #print (bin(temp_c))

    # Convert to 2s complement (temperatures can be negative)
    temp_c = twos_comp(temp_c, 12)

    # Convert registers value to temperature (C)
    temp_c = temp_c * 0.0625

    # convert the temp to a pattern of YY.XX
    return round(temp_c, 2)

##############################################################################
# SSD Function - separate the value into two parts. each one per one digit
```

```python

def get_digits(NUM):
    return [NUM//10 , NUM%10]


def TMP_FUNC (Button):
    global flag
    flag =True
    time.sleep(0.4)
    print ("HI")
    TMP_RESULT = read_temp()

    INT_TMP = TMP_RESULT // 1
    DEC_TMP = TMP_RESULT % 1

    [INT1,INT2] = get_digits(INT_TMP)
    [DEC1,DEC2] = get_digits(DEC_TMP)

    digits=[INT1 ,INT2 ,DEC1 ,0x43]  #0x43 is "C" character
    time.sleep(0.1)
#   We want to clear the SSD first
    bus.write_byte(SSD_ADDRESS,0x76)

    time.sleep(0.1)
#   Move the cursor mode
    bus.write_byte(SSD_ADDRESS,0x79)

    time.sleep(0.1)
#   Move cursor to the first digit
    bus.write_byte(SSD_ADDRESS,0)

    for dig in digits:

        time.sleep(0.1)
        bus.write_byte(SSD_ADDRESS,int(dig))
        print(int(dig))
```

```
    #Those are responsible for printing "." on the SSD
    bus.write_byte(SSD_ADDRESS, 0x77)
    bus.write_byte(SSD_ADDRESS, 0x02)

    time.sleep(2)
    #bus.write_byte(SSD_ADDRESS,0x76)
    flag = False


##########################################
########## Initializations###############
##########################################

# Button SETUP
Button = 15
GPIO.setmode(GPIO.BOARD)
GPIO.setup(Button,GPIO.IN,pull_up_down=GPIO.PUD_UP)
print("we were here")
global flag
flag = False
# Our bus belong to channel 1 of the I2c Protocol(since 2014)
i2c_ch = 1

# TMP102 address on the I2C bus
TMP_ADDRESS = 0x48
time.sleep(0.1)
# SSD address on the I2C bus
SSD_ADDRESS= 0x77
# Register addresses  (Belong to TMP102)
reg_temp = 0x00
reg_config = 0x01

# Initialize I2C (SMBus)
time.sleep(0.1)
bus = smbus.SMBus(i2c_ch)
time.sleep(0.1)
```

```
# Read thep[ CONFIG register (2 bytes)
val = bus.read_i2c_block_data(TMP_ADDRESS, reg_config, 2)
print("Old CONFIG:", val)

# Set to 4 Hz sampling (CR1, CR0 = 0b10)
val[1] = val[1] & 0b00111111
val[1] = val[1] | (0b10 << 6)

# Write 4 Hz sampling back to CONFIG
bus.write_i2c_block_data(TMP_ADDRESS, reg_config, val)

# Read CONFIG to verify that we changed it
val = bus.read_i2c_block_data(TMP_ADDRESS, reg_config, 2)
print("New CONFIG:", val)


#######################################################
#####################Main Code##########################
#######################################################

# If the button is pressed
GPIO.add_event_detect(Button,GPIO.BOTH,callback=TMP_FUNC)

while True:
    print("im in this loop")
    if (not(flag)):

        # [3] position has the hours value
        current_hours=time.gmtime()[3]
        # [4] position has the minutes value
        current_min=time.gmtime()[4]
        # Let's separate the values into two digits
        [h1,h2]=get_digits(current_hours)
        # The Hours value is not suitable with Israel clock, diffrence of 3
        h2=h2+3
        current_min=time.gmtime()[4]
```

```
        [m1,m2]=get_digits(current_min)

        time.sleep(0.1)
    #   Move the cursor mode
        bus.write_byte(SSD_ADDRESS,0x79 )

        time.sleep(0.1)
    #   Move cursor to the first digit
        bus.write_byte(SSD_ADDRESS,0)

        # burn the digits into the SSD
        digits=[h1 ,h2 ,m1,m2]

        for dig in digits:

            time.sleep(0.1)
            bus.write_byte(SSD_ADDRESS,int(dig))
        #Those are responsible for printing ":" on the SSD
        bus.write_byte(SSD_ADDRESS, 0x77)
        bus.write_byte(SSD_ADDRESS, 0x10)

```

ההסברים לתוכנה מופיעים בגוף הקוד עצמו. במקום שאין הסברים זה מכיוון שהקוד חוזר על עצמו.

סיכום

למדנו להתגבר על הרבה מכשולים בדרך, כאלו שקשורים להבנת מרכיביו ותכונותיו של כל רכיב, וכאלו שקשורים לאופן כתיבת הקוד. בניית המעגל הייתה מאוד מהירה כיוון ששרטטנו לפני כן כמו שצריך את המעגל שאנחנו היינו אמורים לבנות.

התרגשנו כאשר הפרויקט שלנו הצליח. לראות כיצד חלום מתגשם לו. המחשבה שיום יבוא וישתמשו בהמצאה שלנו (שעון יד עם מד טמפ') חיממה לנו את הלב.