תקציר

תכנון ומימוש תיבת דואר חכמה המונה את כמות המכתבים ומוגנת באמצעות קודן כניסה.
המערכת הורכבה ממספר חלקים עיקריים: חיישן Ultrasonic, צג 16 ביט, מנוע, זמזם וקודן בן 4 ספרות.
המערכת מומשה באמצעות בקר Raspberry pi 3 ותוכנתה בשפת Python.
מטרת הפרויקט היא תכנון ומימוש של תיבת דואר חכמה כאשר מודגמת שליטה על חומרה באמצעות תוכנה.
בעת הכנסת דואר מתקיימים השלבים הבאים:

1. הכנסת דואר לתיבה – צג מורה על כמו המכתבים שהוכנסה.
2. הוצאת דואר מן התיבה – הקשת סיסמה ופתיחת הדלת, בעת הקשה לא נכונה יופעל זמזם.
3. שחרור נעילת הדלת – בעת הקשת סיסמה נכונה יופעל מנוע לשחרור הנעילה.

תודות

ברצוננו להודות למרצה הקורס פרופסור אהרון פרידמן שעורר בנו את הרצון להצליח ודחף לכך שנוציא המיטב מהידע שנרכש במהלך התואר. על הסבלנות והעזרה מרחוק, במיוחד בתקופה מאתגרת זו.

תוכן עניינים

1.מבוא

1.1 מקור המושג

המילה "דואר" מקורה ככל הנראה בשפה הארמית,
"אין משלחין... הא דקביע בי דואר במתא והא דלא קביע בי דואר במתא". פירוש רבינו חננאל מסביר את המונח בי דואר במילים "איש ידוע, שכל כתב אליו יובל, והוא המשכיר ומשלח כל איגרת למי שנשלחה לו". רש"י מסביר את המונח במילים "שלטון העיר ולו רגילין לשלוח איגרות".
היא משקפת את המונח "דוּר" שמשמעותו "סיבוב". מנוח זה מאפיין את צורת פעילותם של עובדי שירות הדואר, אשר להבדיל מן השליח הנושא את המכתב כל הדרך מן השולח אל המכותב, פועלים בסבבים בהם כל אחד מן המעורבים אחרי לטיפול במכתב במקטע אחד של הדרך עד למסירתו לפקיד הבא בתהליך.

1.2 התפתחות תיבות הדואר

תיבת הדואר הראשונה שהוצגה לשירות הציבור הוצבה בפירנצה ב-1650 והייתה מיועדת להנחת מכתבי הלשנה אנונימיים. בשנת 1653 הוצבו תיבות דואר ברחבי פריז, באיים הבריטיים הוצבה תיבת דואר עצמאית (כזו שאינה חלק ממבנה, אלא מוצבת ברחוב) באי ג'רזי בשנת 1852 וזאת בהמלצת הסופר אנתוני טרולופ, שבאותה העת עבד בדואר.
שנה לאחר מכן הותקנה תיבה כזו באנגליה. בארצות-הברית התוקנו תיבות דואר בשנות ה-50 של המאה ה-19.
הדואר המודרני מנוהל על ידי ארגונים לאומיים ופרטיים כאחד, המקושרים זה לזה באמצעות תקנות והסכמים בינלאומיים. בעוד שהמכתבים, שהיו פופולריים בעבר, הוחלפו ברובם על ידי אמצעי תקשורת אלקטרוניים, נעשה שימוש במערכת הדואר גם להעברת מכתבים רגילים.

1.3 הצורך

תיבת דואר חכמה היא הכרחית ע"מ לדעת כמה מכתבים התיבה מכילה ושמירה על פרטיות תכולת תיבת הדואר באמצעות קודן.

2. מטרת הפרויקט

פיתוח ותכנון תיבת דואר חכמה המדגימה שליטה על חומרה באמצעות תוכנה.

2.1 תנאי תכנון

תנאי המערכת:
נעילת דלת – ללא נעילת הדלת התיבה לא תמנה מכתבים.
הקשת קודן – במתן סיסמה נכונה\שגויה הדלת תינעל\תיפתח בהתאמה.
פעולת המנוע – בעת הקשת קודן נכון יופעל המנוע לפתיחה וסגירה של הבריח.
זמזם – בעת הקשת קודן שגוי יופעל הזמזם.
צג דיגיטלי – צג המורה על כמות מכתבים ועל הקשת סיסמה נכונה\שגויה.

2.2 דרישות מערכת

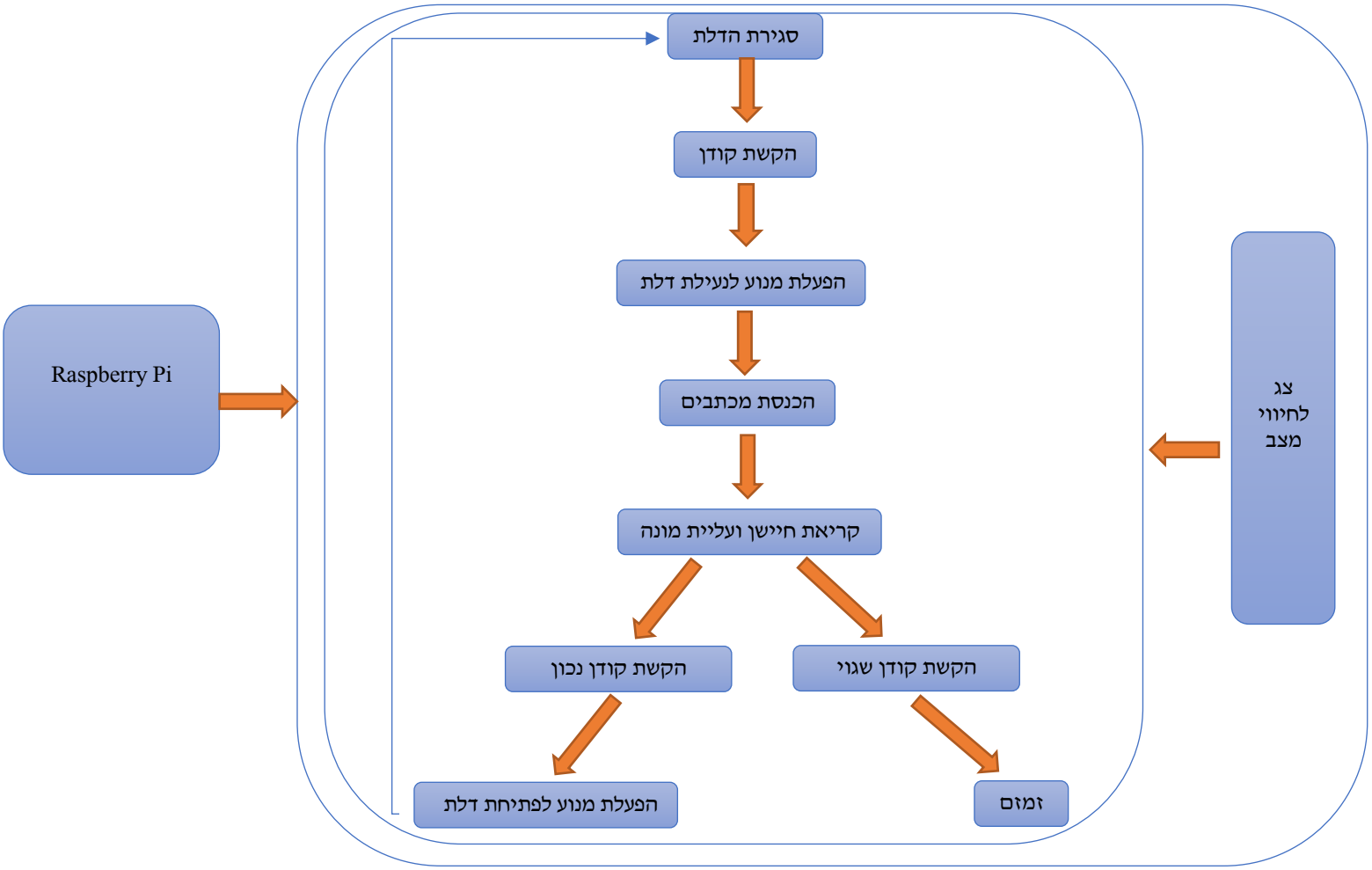
תיבת הדואר תתריע על כמות המכתבים בתיבה ותהיה מוגנת באמצעות קודן.
כאשר במשך כל התהליך ישנו צג המורה על כמות המכתבים ועל הקשת סיסמה נכונה\שגויה.
בעת הקשת סיסמה שגויה יופעל זמזמם, בעת הקשת סיסמה נכונה תפתח נעילת הדלת באמצעות מנוע.

3. תיאור המערכת

מערכת תיבת הדואר החכמה מונה מכתבים ומורה על הקשת סיסמה נכונה תוך הצגת המידע על גבי הצג הדיגיטלי

3.1 הסבר כללי על המערכת

סכמת בלוקים:



איור – 1 סכמת בלוקים של כלל המערכת.

פירוט סכמת הבלוקים:

1. Raspberry pi3 : בקר האחראי על תפעול המערכת.
2. צג לחיווי מצב : תפקידו לאפשר למבצע חיווי בעבור כמות המכתבים ואינדיקציה על קודן נכון\שגוי ומצב הדלת.
3. סגירת הדלת : בעת סגירת הדלת יופיע על הצג בקשה להקשת סיסמה.
4. הפעלת מנוע : כאשר הדלת סגורה ולאחר נעילת הדלת ע"י קודן יופעל המנוע לנעילת הבריח.
5. קריאת חיישן ועליית מונה : תפקידו של החיישן הוא העלאת ערך המנייה בעת הכנסת מכתב וזאת כאשר הדלת סגורה ונעולה.
6. קודן : הקשת קודן לפתיחה או נעילת של הדלת כאשר הדלת סגורה.
7. זמזם : תפקידו להשמיע צליל בתדר גבוה לאחר הקשת קודן שגוי.
8. הפעלת מנוע לפתיחת הדלת : בעת הקשת קודן נכון המונע נכנס לפעולה ופותח את נעילת הבריח.

3.2 תכנון החומרה

פירוט רכיבים:
צג 16bit דיגיטלי (1602)
צג דיגיטלי המכיל עד 16 סיביות בכל שורה ובעל שתי שורות.

חיישן מרחק (HC-SR04)
חיישן בעל רכיב שידור ורכיב קליטה, המשדר תדר באמצעות רכיב השידור ומודד את מרחק ההחזרה באמצעות רכיב הקליטה ע״י חישב זמן ההחזרה מן האובייקט.

מנוע Servo (SG90)
מנוע servo הוא מנוע זרם ישר (DC Motor) בעל מערכת תמסורת פנימית של גלגלי שיניים ובקרה אלקטרונית על מיקום המנוע.
מה שמייחד מנועי servo היא העובדה שהם אינם מסתובבים בצורה חופשית כמו מנועי DC, אלא נעים על פי זווית – לרוב בין 0 ל- 180 מעלות.
מנועי servo פועלים בחוג סגור, כלומר הינם בעלי בקרה על מיקום המנוע, ובעלי יכולת תיקון פערים מהמיקום הרצוי.

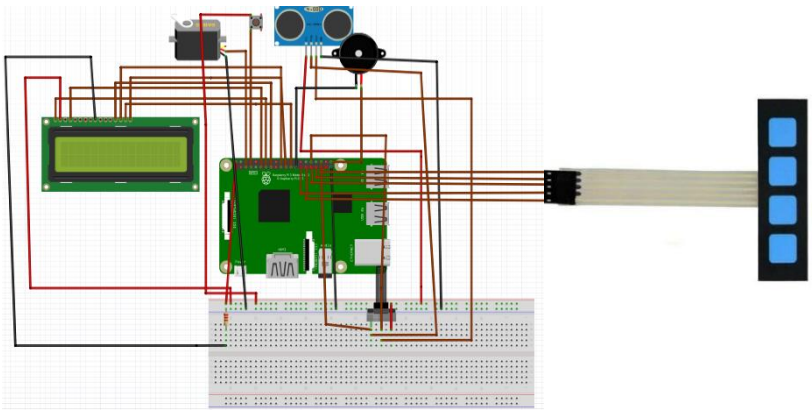
לוח מקשים 4x1
הקודן מורכב מארבע ספרות, כל כפתור מרגיש כמו בועת אוויר קטנה.

זמזם (Generic buzzer)
רכיב זה מופעל במתח חיובי של ׳1׳ לוגי ומפיק צפצוף ונכבה במתח של ׳0׳ לוגי.

3.3 תכנון תוכנה

אלגוריתם התוכנה נכתב ומומש באמצעות שפת Python. נעשה שימוש בספריות חיצוניות לחלק מן הפונקציות, כאשר ישנה פונקציה ראשית אחת שמריצה את כל הפונקציות המשניות וקוראת להן.

4.מימוש המערכת



איור – 2 סכמת חיבורים של המערכת.

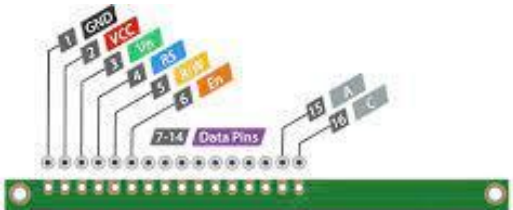
איור 3 מימוש המערכת בפועל

4.1 מימוש חומרה

צג 16bit דיגיטלי (1602)

צג בעלת 16 סיביות ושתי שורות, החיבורים העיקריים בהם נעשו שימוש הם:

- Vss – אדמה
- Vdd – מתח הזנה
- V0 – אדמת רפרנס לקביעת בהירות התצוגה
- RS – בחירת הרגיסטר הרלוונטי (שליחת פקודות או מידע)
- R\W – קריאה או כתיבה למסך
- E – כניסת אפשור
- DB4-DB7 – כניסת מידע המאפשרת שימוש ב-4 ו-8 סיביות
- BLA – מתח תאורה אחורית
- BLK – אדמת רפרנס לתאורה אחורית



איור – 4 מסך LCD.

חיישן מרחק (HC-SR04)

החיישן מורכב משני חלקים עיקריים:

1. חיישן המשדר תדר
2. חיישן הקולט את התדר המוחזר

החיבורים המשמשים אותנו במהלך פרויקט זה הם:

- Vcc – מתח הזנה
- Trig – כניסת בקרה השולטת בהפעלת חיישן השידור
- Echo – יציאת מידע של החיישן אשר קולט את המידע המוחזר מן האובייקט
- GND – אדמה

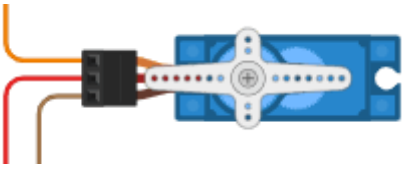
איור – 5 חיישן Ultrasonic.

מנוע Servo (SG90)

מנועי Servo משמשים ברובוטיקה, טיסנים מכוניות על שלט או כל שימוש אחר שמצריך דיוק מסוים מבחינת מיקום המנוע. הרעיון הוא שלמנוע יש פידבק שמציין את המיקום שלו ביחס לנקודה ההתחלתית. Servo טיפוסי יכול לנוע רק בטווח של 180 מעלות אבל ישנם גם כאלה שזים סיבוב מלא.

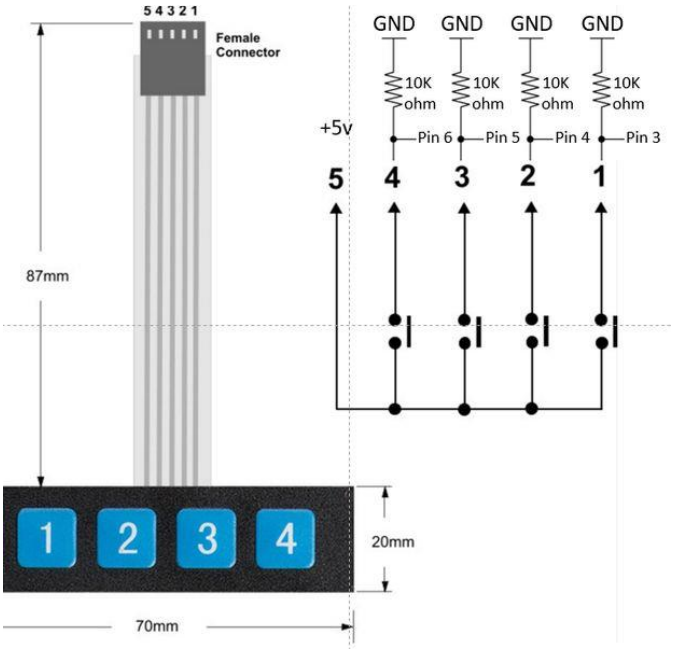
החיבורים הקיימים במנוע הם:

- מתח הזנה של 5V
- כניסת בקרה לשליטה במנוע
- אדמה



איור – 6 מנוע Servo.

לוח מקשים 4x1:
לוח המקשים חובר לפיני הבקרה בבקר ולמתח קבוע של 5V.
בזמן לחיצה מתרחש "קצר" ע"י קונפיגורציה של פיני הבקרה נוכל לדעת איזה מספר נלחץ בפועל.



איור – 7 חיווט פנימי של לוח המקשים.

זמזם (Generic buzzer):

שימוש בזמזם אקטיבי, אשר מתאפיין בקבלת אורכי פולסים שונים ע"מ ליצור צליל.

איור – 8 זמזם.

4.2 מימוש תוכנה

יובאו מספר ספריות, חלקם הורדו מהאינטרנט וחלקם נוצרו ידנית

```
import RPi.GPIO as GPIO
from pad4pi import rpi_gpio as RPG
import units.settings as settings
from units.settings import defaultPassword as password
from units.door import Door
from units.lcd import lcd, clearLine, writeLine
from units.utils import DetectChange
from units.latter import LatterCounter, LatterDetector
import time
```

נעשה אתחול ראשוני לרכיבים ולפונקציות

```
GPIO.setup(settings.pins["door"], GPIO.IN,
pull_up_down=GPIO.PUD_DOWN) # init door 1 on ,0/null off
GPIO.setup(settings.pins["buzzer"], GPIO.OUT) # init buzzer pin
buzzer = GPIO.PWM(settings.pins["buzzer"], settings.buzzerFreq)
buzzer.start(0) # duty cycle=0


#init kaypad ,init 4 buttons 1,2,3,4,the kaypad consist of a
combination of rows and columns
keypad = RPG.Keypad(keypad=[[1, 2, 3, 4]],
row_pins=settings.pins["keypad"]["rows"],
col_pins=settings.pins["keypad"]["cols"])
doorLock = Door(settings.pins["servo"], settings.servoFreq) #init
object for the servo
doorDetector = DetectChange() #init object with which we will check
if there has been a change in the opening and closing of the door

lcd.clear() # clear lcd
writeLine(0, '') #write null lcd  because invalid valuse
latterDetector = LatterDetector() # init num of letter in mail box =0
```

לאחר מכן נבדק האם הדלת פתוחה או סגורה.

כאשר הדלת סגורה ונעולה, המשתמש מתבקש להקיש סיסמה על מנת לפתוח את הנעילה. אם הקוד שגוי נרשמת על המסך הודעת שגיאה ונשמע צליל הזמזם.

כאשר הדלת סגורה אך אינה נעולה, המשתמש מתבקש להקיש סיסמה על מנת לנעול את הדלת.

```
global password, currentPass
  doorClosed = GPIO.input(settings.pins["door"])
  if doorClosed:
      if currentPass == 0: # if dor is lock and no pasword
          writeLine(1, 'password:      ') # write on lcd on row 1
          lcd.cursor_pos = (1, 10)
      lcd.write_string('*')
      currentPass = currentPass * 10 + key
```

```
        if 1000 <= currentPass < 10000: # the password  contains 4
digit
            if not doorLock.isLock(): # if the door dont lock
                password = currentPass
                time.sleep(.5)
                doorLock.lock()
                lcd.clear()
                writeLine(0, 'Door locked')
                writeLine(1, 'successfully!')
                time.sleep(1.5)
                lcd.clear()
                writeLine(0, 'Number of')
                writeLine(1, ('latters: %i') %
latterDetector.getLatterCounter().getCounter())
            else: # the door is lock
                if password == currentPass: # the door is lock and
the passowrd is true
                    time.sleep(.5)
                    doorLock.unlock()
                    lcd.clear()
                    writeLine(0, 'Door unlocked')
                    writeLine(1, 'successfully.')
                    time.sleep(1.5)
                    lcd.clear()
                    writeLine(0, 'Take out')
                    writeLine(1, 'the latters')
                else:# the door is lock and the passowrd is not true
                    time.sleep(.5)
                    lcd.clear()
                    writeLine(0, 'Wrong password!')
                    for _ in range(3):
                        buzzer.ChangeDutyCycle(50)
                        time.sleep(1)
                        buzzer.ChangeDutyCycle(0)
                        time.sleep(1)
                        lcd.clear()
                        writeLine(0, 'Number of')
                        writeLine(1, ('latters: %i') %
latterDetector.getLatterCounter().getCounter())

            currentPass = 0
```

ישנה מתודה בשם doorChanged אשר תפקידה לבדוק האם היה שינוי בפתיחה או סגירה של הדלת.

מתודה זאת נוצרה על מנת לבצע אינדיקציה האם הדלת נפתחה או נסגרה.

```
def doorChanged(pin):
    global currentPass
    result, doorClosed = doorDetector.detect(GPIO.input(pin))
    if result and not doorClosed: # if the door is open ,rest the
counter letter
        latterDetector.getLatterCounter().clearCounter()
    if result and not doorLock.isLock(): # if detect door and the
soor is not lock
        currentPass = 0
        if doorClosed:
            lcd.clear()
            writeLine(0, 'Enter password')
            writeLine(1, 'to lock the door.')
        else:
```

```
            lcd.clear()
            writeLine(0, 'Mail box:')
            writeLine(1, 'Close the door.')
```

לוח המקשים עובד כפסיקה ולכן נעשתה הגדרה בקוד ולאיזה מתודה הפסיקה קופצת

```
keypad.registerKeyPressHandler(keypress) #input from keypad
GPIO.add_event_detect(settings.pins["door"], GPIO.BOTH,
callback=doorChanged) # event to the pin of door and up to the
function doorChanged
```

לאחר מכן הוגדר מספר הפין אשר תפקידו לקבל 0 או 1 בהתאם למצב הדלת

```
doorChanged(settings.pins["door"])
```

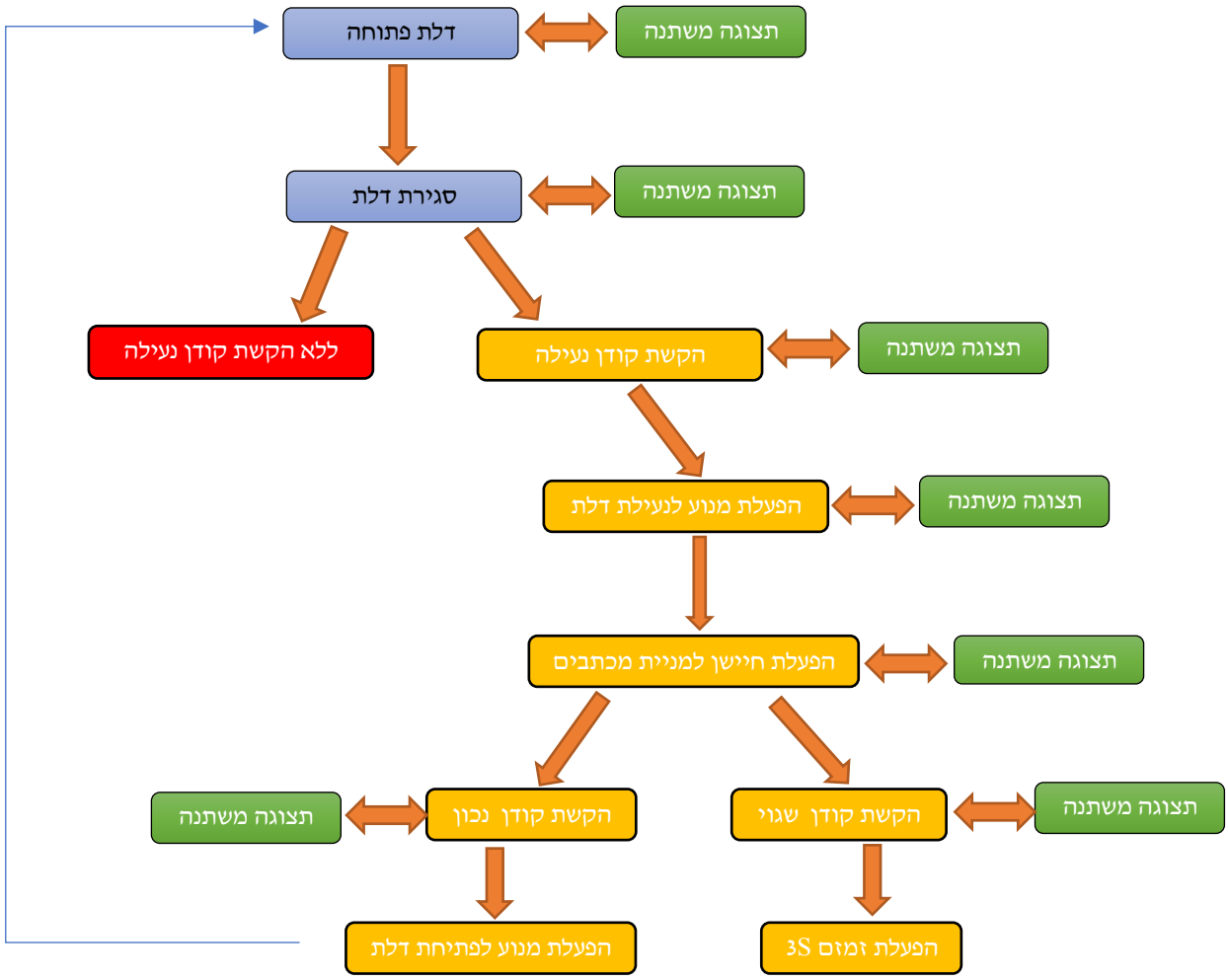
לבסוף הוגדר כאשר הדלת סגורה ונעולה, ישנה ספירה אינסופית של מכתבים עד לפתיחת הנעילה והדלת.

```
try:
    while True:
        if latterDetector.detect() and doorLock.isLock(): # if the
door lock and the Detect of the door
            lcd.clear()
            writeLine(0, 'Number of')
            writeLine(1, ('latters: %i') %
latterDetector.getLatterCounter().getCounter())

except KeyboardInterrupt:
    GPIO.cleanup()
```

4.2.1 אלגוריתם התוכנה

סכמת בלוקים



איור – 9 סכמת אלגוריתם התוכנה.

5.תקלות ופתרונות

כיול חיישן – התגלתה בעיה של כיול המרחק ע״מ ליצור איטרציה במרחק הנכון. היה עלינו לכייל את החיישן למרחק המתאים של פתח הכנסת הדואר לתיבה כדי שסף המרחק יצור התראה בחיישן. לפני פתרון הבעיה, מכתב שהיה מוכנס לתיבה לא היה נקרא בחיישן מכיוון שסף המרחק היה מכויל לחלק התחתון ביותר של התיבה.

מיקום ספרות – סדר מיקומם הפיזי של הספרות בלוח המקשים שונה מאשר הסדר בו הם נקראים ע״י התוכנה.
ע״מ לפתור בעיה זו בוצעה הצבה מחדש של כל ספרה בתוך משתנה חדש המעיד על מיקומה הפיזי של הספרה.

חוסר יציבות הServo – מנוע הServo מאוד רגיש לשינויים בזרם, ועקב זרם לא יציב במערכת נוצרו ריצודים בתנועת המנוע לעומת מצב סטטי ללא תזוזה. בעיה זו לא נפתרה מכיוון שהיא אינה מהווה חסרון בפעולת המערכת.

קוטביות הפוכה בצג – צג הLCD חובר בטעות בקוטביות הפוכה לפינים האחראיים על בהירות התצוגה, מעשה זה גרם לקצר ושריפת הצג. בעיה זו נפתרה ע״י בדיקה של כל פין ופין היוצא מן הבקר ולבסוף חיבור צג חדש.

6.סיכום ומסקנות

התבקשנו לתכנן ולפתח מערכת שבה ניתן להדגים שליטה על חומרה באמצעות תוכנה כאשר המערכת מבוססת על בקר Raspberry Pi 3 והאלגוריתם נכתב בשפת Python.
לאחר תכנון ראשוני של סכמת בלוקים ופירוק רכיבי המערכת העיקריים לכל בלוק בנפרד, התחלנו בתכנון ומימוש המערכת ע״י רכיבים דיסקרטיים כאשר השלב הראשוני היה קריאת הרכיבים ובחלק מן הפעמים ע״י שימוש בAPI. לאחר מכן התחלנו בכתיבת פונקציות ומחלקות כדי שבסוף נוכל לאחד אותם אל פונקציה ראשית כדי לקצר את זמן הריצה של המערכת ולחסוך במשאבי זיכרון.
אילולא בעיות של זמן וכסף, היינו ממשיכים בייעול של חלקי המערכת ובהוספת פיצ׳רים, לדוגמה:

- שימוש בחיישן מבוסס משקל למניית מכתבים
- הטמעת מערכות התראות לטלפון הסלולרי בעת קבלת מכתב
- שדרוג גודל מסך התצוגה להכלה של יותר טקסט

7.ביבליוגרפיה

Bargen, D. (n.d.). *RPLCD.* Retrieved from https://rplcd.readthedocs.io/en/stable/api.html#charlcd-gpio

GPIO. (n.d.). Retrieved from raspberrypi: https://www.raspberrypi.org/documentation/usage/gpio/

Heywood. (n.d.). *Raspberry Pi Python Library for Ultrasonic Module HC-SR04 Distance Measuring Transducer.* Retrieved from pypi: https://pypi.org/project/Bluetin-Echo/

https://docs.python.org/3/library/time.html. (n.d.). *time — Time access and conversions.* Retrieved from https://docs.python.org/3/library/time.html

https://learn.sparkfun.com/tutorials/raspberry-gpio/python-rpigpio-api. (n.d.). *Raspberry gPIo.* Retrieved from sparkfun: https://learn.sparkfun.com/tutorials/raspberry-gpio/python-rpigpio-api

pad4pi 1.1.5. (n.d.). Retrieved from Python Software Foundation: https://pypi.org/project/pad4pi/

Raspberry Pi Servo Motor control. (n.d.). Retrieved from tutorials-raspberrypi: https://tutorials-raspberrypi.com/raspberry-pi-servo-motor-control/

גונן, ב'. (2021). *תכנות בשפת פייתון*. אוחזר מתוך https://data.cyber.org.il/python/python_book.pdf

8.רשימת איורים.

9.נספחים.

:main

```
import RPi.GPIO as GPIO
from pad4pi import rpi_gpio as RPG
import units.settings as settings
from units.settings import defaultPassword as password
from units.door import Door
from units.lcd import lcd, clearLine, writeLine
from units.utils import DetectChange
from units.latter import LatterCounter, LatterDetector
import time

currentPass = 0

GPIO.setup(settings.pins["door"], GPIO.IN,
pull_up_down=GPIO.PUD_DOWN) # init door 1 on ,0/null off
GPIO.setup(settings.pins["buzzer"], GPIO.OUT) # init buzzer pin
buzzer = GPIO.PWM(settings.pins["buzzer"], settings.buzzerFreq)
buzzer.start(0) # duty cycle=0


#init keypad ,init 4 buttons 1,2,3,4,the keypad consist of a
combination of rows and columns
keypad = RPG.Keypad(keypad=[[1, 2, 3, 4]],
row_pins=settings.pins["keypad"]["rows"],
col_pins=settings.pins["keypad"]["cols"])
doorLock = Door(settings.pins["servo"], settings.servoFreq) #init
object for the servo
doorDetector = DetectChange() #init object with which we will check
if there has been a change in the opening and closing of the door

lcd.clear() # clear lcd
writeLine(0, '') #write null lcd  because invalid values


latterDetector = LatterDetector() # init num of letter in mail box =0

def keypress(key):
    global password, currentPass
    doorClosed = GPIO.input(settings.pins["door"])
    if doorClosed:
        if currentPass == 0: # if door is lock and no password
            writeLine(1, 'password:     ') # write on lcd on row 1
            lcd.cursor_pos = (1, 10)
        lcd.write_string('*')
        currentPass = currentPass * 10 + key
        if 1000 <= currentPass < 10000: # the password contains 4
digit
            if not doorLock.isLock(): # if the door won`t lock
                password = currentPass
```

```
                time.sleep(.5)
                doorLock.lock()
                lcd.clear()
                writeLine(0, 'Door locked')
                writeLine(1, 'successfully!')
                time.sleep(1.5)
                lcd.clear()
                writeLine(0, 'Number of')
                writeLine(1, ('latters: %i') %
latterDetector.getLatterCounter().getCounter())
            else: # the door is locked
                if password == currentPass: # the door is locedk and
the password is true
                    time.sleep(.5)
                    doorLock.unlock()
                    lcd.clear()
                    writeLine(0, 'Door unlocked')
                    writeLine(1, 'successfully.')
                    time.sleep(1.5)
                    lcd.clear()
                    writeLine(0, 'Take out')
                    writeLine(1, 'the latters')
                else:# the door is locked and the password is not
true
                    time.sleep(.5)
                    lcd.clear()
                    writeLine(0, 'Wrong password!')
                    for _ in range(3):
                        buzzer.ChangeDutyCycle(50)
                        time.sleep(1)
                        buzzer.ChangeDutyCycle(0)
                        time.sleep(1)
                        lcd.clear()
                        writeLine(0, 'Number of')
                        writeLine(1, ('latters: %i') %
latterDetector.getLatterCounter().getCounter())

            currentPass = 0

def doorChanged(pin):
    global currentPass
    result, doorClosed = doorDetector.detect(GPIO.input(pin))
    if result and not doorClosed: # if the door is open, reset the
letter counter
        latterDetector.getLatterCounter().clearCounter()
    if result and not doorLock.isLock(): # if detect door and the
door is not lock
        currentPass = 0
        if doorClosed:
            lcd.clear()
            writeLine(0, 'Enter password')
            writeLine(1, 'to lock the door.')
        else:
            lcd.clear()
            writeLine(0, 'Mail box:')
            writeLine(1, 'Close the door.')


keypad.registerKeyPressHandler(keypress) #input from keypad
```

```
GPIO.add_event_detect(settings.pins["door"], GPIO.BOTH,
callback=doorChanged) # event to the pin of door and up to the
function doorChanged

doorChanged(settings.pins["door"])

try:
    while True:
        if latterDetector.detect() and doorLock.isLock(): # if the
door is locked and the detect of the door
            lcd.clear()
            writeLine(0, 'Number of')
            writeLine(1, ('latters: %i') %
latterDetector.getLatterCounter().getCounter())

except KeyboardInterrupt:
    GPIO.cleanup()
```

Door:

```
from units.servo import Servo
import units.settings as settings

class Door:
    def __init__(self, pin, freq): #init the door
        self.__door = Servo(pin, freq) #object on the servo
        self.unlock()# unlock the door

    def isLock(self):
        return self.__locked #return if the door is locked or
unlocked

    def lock(self): # the -90 angels the door is unlocked - true
        self.__door.setAngle(-90)
        self.__locked = True

    def unlock(self): # the 0 angels the door is unlocked - false
        self.__door.setAngle(0)
        self.__locked = False
```

latter:

```
import units.sonar as sonar
import units.settings as settings

class LatterCounter:
    def __init__(self): # init  counter
        self.__counter = 0

    def getCounter(self): # return num of letter on box
        return self.__counter

    def increaseCounter(self): # up counter on the box
        self.__counter += 1

    def clearCounter(self):
        self.__counter = 0
```

```
class LatterDetector:
    def __init__(self):
        self.__prevDist = 0
        self.__letterCounter = LatterCounter()

    def getLatterCounter(self):
        return self.__letterCounter

    def latterInserted(self, dist):
        latterSettings = settings.height["latter"] # get setting  min
and max distance
        return latterSettings["min"] <= dist < latterSettings["max"]
# if the distance between min to max than into letter in mailbox

    def detect(self):
        dist = sonar.readDist() #get distance
        print(dist)
        detected = self.latterInserted(dist) and not
self.latterInserted(self.__prevDist) #get there was inserted
        if detected:
            self.__letterCounter.increaseCounter() #counter up
        self.__prevDist = dist #now dis is the prevDist on the next
distance
        return detected #return the distance
```

lcd:

```
import RPi.GPIO as GPIO
from RPLCD import CharLCD
import units.settings as settings

lcd = CharLCD(numbering_mode = GPIO.BCM, #init lcd
              pin_rs=settings.pins["lcd"]["rs"],
              pin_e=settings.pins["lcd"]["en"],
              pins_data=settings.pins["lcd"]["data"],
              cols=40, rows=2)

def clearLine(line): #clear lcd
    lcd.cursor_pos = (line, 0)
    lcd.write_string(' ' * 40)
    lcd.cursor_pos = (line, 0)

def writeLine(line, msg): #write on the lcd
    lcd.cursor_pos = (line, 0)
    msg += (' ' * (40 - len(msg)))
    lcd.write_string(msg)
```

servo:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

#the servo work on 180 angles and we uses offset 90 angels from -90
angels to 90 angles

class Servo:
    def __init__(self, pin, freq):
        GPIO.setup(pin, GPIO.OUT)
        self.__angle = 0
        self.__servo = GPIO.PWM(pin, freq)
```

```
        self.__servo.start(self.__getDutyFromAngle())

    def __getDutyFromAngle(self): #the servo work with duty cycle
change, remote angle to the duty cycle
        return self.__angle * 5 / 90 + 7.5



    def setAngle(self, angle):
        self.__angle = angle
        self.__servo.ChangeDutyCycle(self.__getDutyFromAngle())
#time.sleep(0.5)
```

settings:

```
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
#*************** digit pin**********


sonar = {#unit [cm]
    "unit": "cm",
    "samples": 3
}

height = {
    "latter": {
        "max": 6,
        "min": 4.5,
    },
    "box": 16
}

pins = {
    "door": 4,
    "servo": 3,
    "buzzer": 21,
    "keypad": {
        "rows": [5],
        "cols": [26, 19, 13, 6],
    },
    "sonar": {
        "echo": 12,
        "trigger": 16
    },
    "lcd": {
        "rs": 17,
        "en": 27,
        "data": [11, 9, 10, 22]
    }
}

servoFreq = 50
buzzerFreq = 1000

defaultPassword = 1234
```

sonar:

```
from Bluetin_Echo import Echo
import units.settings as settings

sonarPins = settings.pins["sonar"] #get the pins for the ultrasonic
sonar = Echo(trigger_pin = sonarPins["trigger"], echo_pin =
sonarPins["echo"], mPerSecond = 343) #init ultrasonic

def readDist():
    return sonar.read(settings.sonar["unit"],
settings.sonar["samples"]) #return distance with average on num of
samples
```

utils:

```
class DetectChange:
    def __init__(self): #init of the door
        self.__prevValue = None

    def detect(self, value): #test if have a detect on the door
        if value != self.__prevValue:
            self.__prevValue = value
            return True, value
        else:
            return False, None
```