2024

Hot pot

תמר שושנה הולצר

325984011

דרכי חנה אלעד

מבוא

אוכל הוא דבר בסיסי בחיינו ש"אם אין קמח אין תורה", מחקרים סטטיסטיים מראים כי אנו מבלים כמות נכבדת מחיינו הבוגרים מול הכיריים לצורך הכנת ארוחות בין אם הארוחות בשבילנו ובין אם הן בשביל כל המשפחה. פוסטים ומאמרים רבים נכתבו על ידי שפים ומדריכים לניהול זמן כיצד לנצל את זמן הבישול לדברים נוספים וזאת מכיוון שמצד אחד זמן הבישול עלול להיות ארוך אך מצד שני הסיר כובל אותנו אליו ואנו חוששים במהלך כל הבישול שמא האוכל ירתח מידי והאוכל יגלוש או לחילופין נשכח את הסיר על האש הוא יתבשל יתר על המידה והאוכל יחמיץ או יישרף.

את הבעיה הזאת החלטתי לפתור במסגרת פרויקט הגמר במערכות משובצות מחשב.

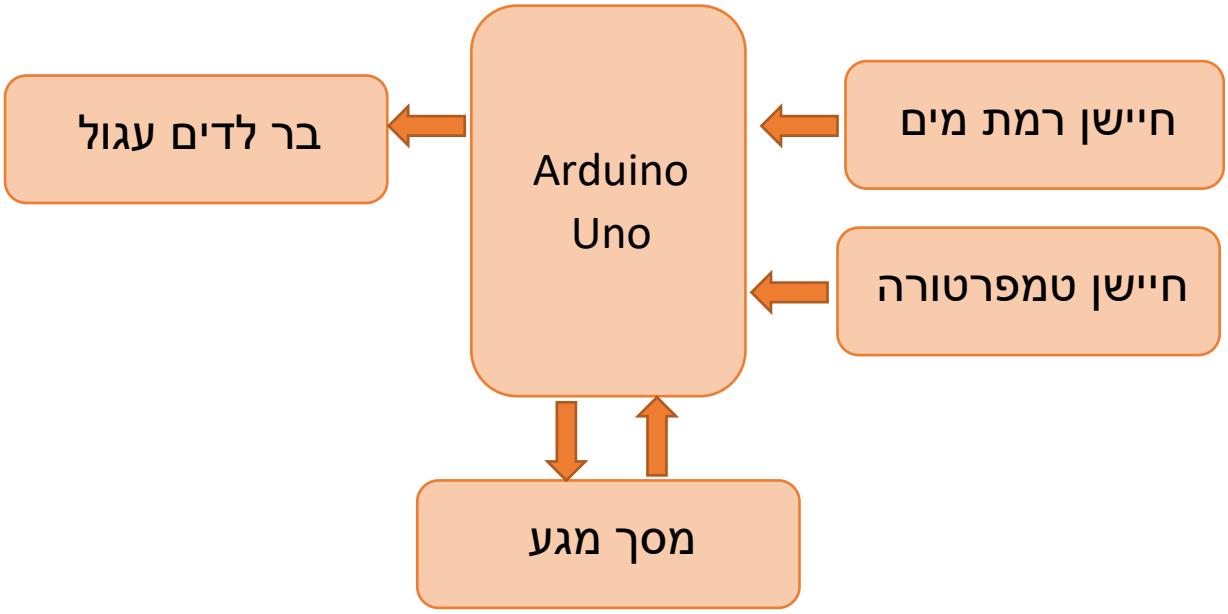
החלטתי לבנות סיר חשמלי, כזה שהמשתמש יבחר על המסך את הטמפרטורה הרצויה ואת משך הזמן שהסיר יפעל. לאחר שהוא יפעיל את הסיר- הסיר ישמור על הטמפרטורה הרצויה יוודא שהתבשיל אינו גולש ולאחר הזמן הרצוי יכבה את הלהבות.

תיאור פעולת המערכת

לסיר החשמלי יהיה מסך מגע אשר שבו המשתמש שרוצה לבשל יבחר את הטמפרטורה הרצויה, במשך הבישול מכשיר יזהה מצב שבו הטמפרטורה עולה על הטמפרטורה הרצויה או יורד ממנה וישנה את צבעי הלהבה בהתאם לשינויי הטמפרטורה וכן יזהה מקרים של בעבוע ועליית מפלס התבשיל גבוה מידי סיכון לגלישה וינמיך את הלהבות בהתאם.

בנוסף לכך המשתמש יבחר במסך את הזמן הקצוב לעמידת התבשיל על האש ולאחר פרק הזמן הרצוי הלהבות יכבו.

תרשים מלבנים



מפרט טכני

- Arduino Uno
- חיישן טמפרטורה LM75
- חיישן רמת מים
- בר לדים עגול
- שעון RTC

טכנולוגיות

חיישן טמפרטורה LM75

הוא חיישן טמפרטורה דיגיטלי הפועל בפרוטוקול I2C בתדר עבודה של עד 400 KHz וכולל מערכת הממירה את הטמפרטורה ממתח אנלוגי למידע דיגיטלי בגודל 11 סיביות עם סימן.

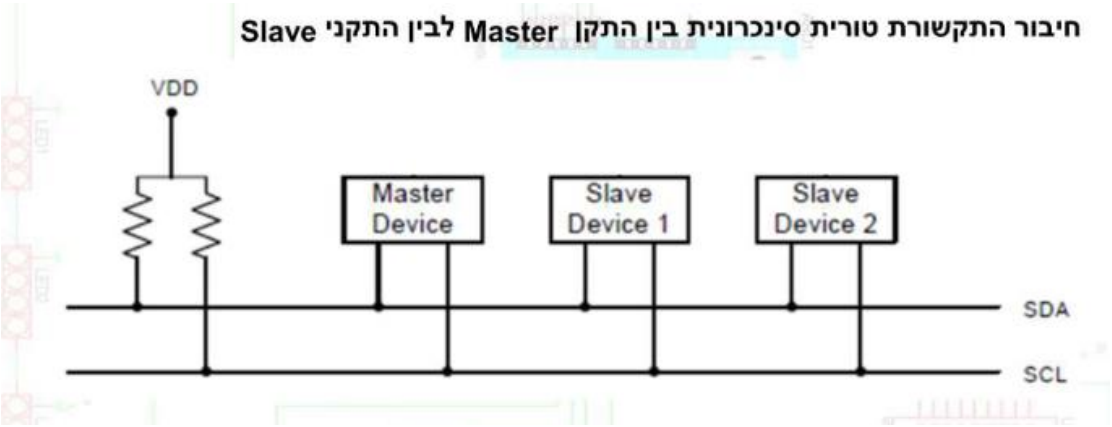
תקשורת טורית i2c – (Inter-Integrated Circuit)

הרכיב מתקשר אל מעבד בתקשורת I2C . על BUS פס תקשורת I2C יכולים להתחבר מספר רכיבים שונים (זיכרונות , ממירים, שעוני זמן אמת וכו'). הרכיב המנהל את תהליך התקשורת (המעבד) נקרא MASTER והרכיבים המתחברים אליו נקראים SLAVES . בתקשורת זו ישנם שני קווים.

קו הנתונים הטורי - SDA - שהוא דו כיווני

וקו השעון הטורי - SCLK שהוא חד כיווני ומופעל על ידי ה MASTER.

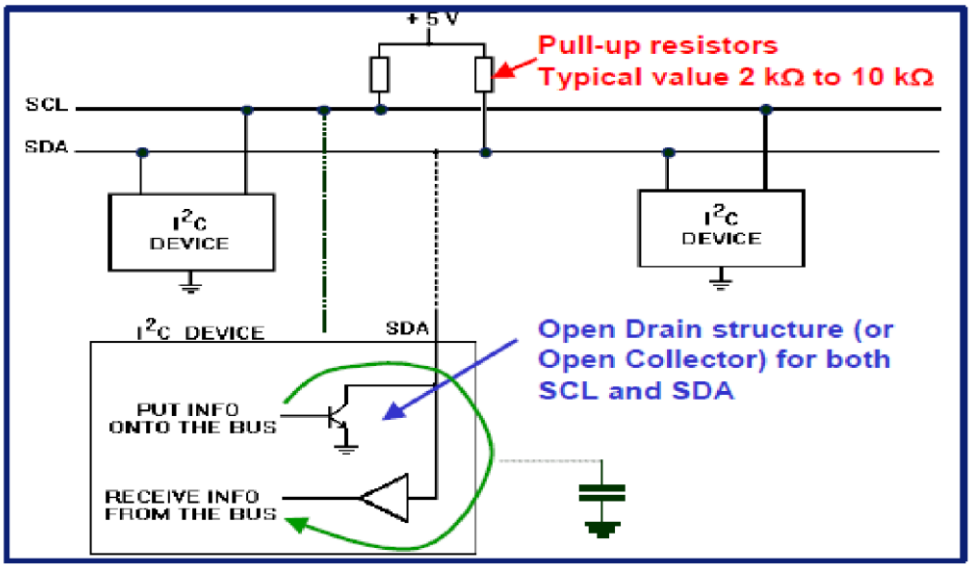
בנוסף, ה MASTER שולט על הגישה לפס ויוצר את מצבי ה -START (התחלה) וה- STOP (סיום).



איור 5 מתאר מספר רכיבים המתחברים על קו התקשורת I2C

איור 5 א' - חיבור של מספר רכיבי SLAVE אל MASTER

באיור 5 א' ניתן לראות 3 רכיבי SLAVE המתחברים אל MASTER. באיור 5 ב' יש פרוט של נגדי ה Pullup וכיצד נראית דרגת היציאה והכניסה של רכיב המתחבר בתקשורת I2C .



איור 5 ב' - קו תקשורת I2C מפורט

ניתן לראות שעל 2 הקווים SDA (קו הנתון) ו SCL (קו השעון) יכולים להתחבר מספר רכיבים . לכל רכיב יש כתובת ייחודית משלו. לרכיב 1307DS הכתובת היא 1101000X (D0H או D1H).

באיור רואים 2 רכיבים המתחברים על הקווים. בחלק התחתון של האיור רואים מבנה פנימי של רכיב ורואים שהרכיב מתחבר בעזרת חוצץ (מתואר על ידי המשולש) המקבל נתון מהקו. מעל החוצץ יש טרנזיסטור בחיבור קולט פתוח

(Open Collector) או טרנזיסטור תופעת שדה - FET - בחיבור מפק פתוח (Open Drain) , שיכול לכתוב לקו נתון.

לטרנזיסטור יש לחבר נגד חיצוני בין 2 קילו אוהם ל 10 קילו אוהם .הערכים נבחרים כך שמצד אחד הנגדים לא יהיו קטנים מידי כדי שלא יזרום זרם גדול דרך הקווים ודרך הרכיב (במצב שהרכיב מוציא 0) ומצד שני שהנגד לא יהיה גדול מידי כי הוא קובע את זמן הטעינה והפריקה במעברים בין 0 ל 1 ולהפך ונגד גדול מידי יגביל את קצב התקשורת.

כללים והגדרות בתקשורת I2C

העברה יכולה להתחיל רק כאשר הקו לא עסוק - NOT BUSY .

- בזמן העברת נתון, קו הנתון חייב להישאר יציב כאשר קו השעון במצב גבוה. שינוי בקו הנתון כאשר קו השעון הוא גבוה יתפרש כאותות בקרה.

מגדירים את מצבי הפס הבאים:

<u>Bus Not Busy -פס לא עסוק</u>

גם קו הנתון וגם קו השעון בגבוה.

<u>START DATA TRANSFER - התחל העברת נתון</u>

שינוי במצב קו הנתון מגבוה לנמוך כאשר השעון נמצא בגבוה מוגדר כמצב START .

<u>STOP DATA TRANSFER - עצור העברת נתון</u>

שינוי במצב קו הנתון מנמוך לגבוה כאשר השעון במצב גבוה מוגדר כמצב STOP .

<u>DATA VALID - תקפות נתון</u>

מצב קו הנתון מייצג תקפות נתון כאשר לאחר מצב START , קו הנתון יציב למשך זמן הגבוה של אות השעון. הנתון בקו חייב להשתנות רק בזמן מצב נמוך של אות השעון. יש פולס שעון אחד עבור כל ביט של נתון.

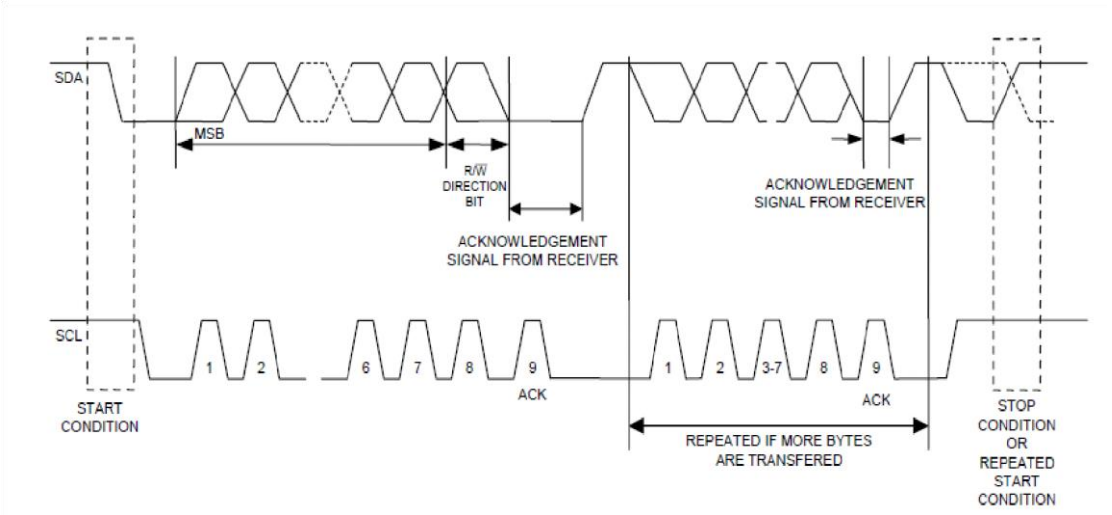
כל העברת נתונים מתחילה עם מצב START ומסתיימת עם מצב STOP . כמות הבתים המועברת בין START ל STOP לא מוגבלת ונקבעת על ידי רכיב ה MASTER . האינפורמציה מועברת ביית אחרי ביית וכל מקלט מאשר קבלת הבית עם ביט תשיעי של ACKNOWLEDGE .
בהגדרות של I2C יש תקן של קצב ב 100KHz ויש תקן ל 400KHz . הרכיב DS1307 עובד בקצב של 100 KHz בלבד.

<u>ACKNOWLEDGE – אישור</u>

כל רכיב קולט חייב בסיום קליטת ביית, שהועבר אליו, ליצור ביט ACKNOLEDGE . רכיב הMASTER יוצר פולס שעון נוסף הקשור לביט זה.

רכיב היוצר ACKNOLEDGE חייב להוריד את קו הנתון הטורי – SDA – ל 0 בזמן פולס השעון ,כלומר שקו הנתון יהיה יציב בנמוך בזמן שקו השעון בגבוה. רכיב ה MASTER מסמן ל SLAVE על סיום התקשורת על ידי אי יצירת ביט ה ACKNOLEDGE כאשר הוא קלט את הביית האחרון מה SLAVE . במקרה כזה על ה SLAVE להשאיר את קו הנתון בגבוה כדי לאפשר ל MASTER ליצור מצב stop.

באיור 6 ניתן לראות העברה של נתון טורי.



איור 6 - העברת נתון בקו תקשורת טורית I2C

את הקו SCL (הקו התחתון בשרטוט) יוצר תמיד ה MASTER. כדאי לשים לב שמצב START קורה כאשר קו SCL בגבוה ואז ה MASTER הוריד את קו הנתון ל 0 . לאחר מכן ה MASTER יוצר 8 פולסי שעון ואז הוא שולח בקו הנתון - SDA - 8 ביטים . 7 ביטים הם כתובת הרכיב והביט ה 8 אומר האם הוא רוצה לכתוב אל הרכיב או לקרא ממנו (0 - כתיבה ,2 – קריאה). לאחר מכן ה MASTER יוצר פולס 9 נוסף שבו ה SLAVE צריך להחזיר ACKNOWLEDGE . לאחר מכן אין צורך ב START נוסף והבתים נשלחים אחד אחרי השני כאשר הצד הקולט נותן ACKNOWLEDGE בביט מספר 9 . מצב STOP)או START חוזר (מתואר בצד ימין של איור 6 . הוא נוצר כאשר קו השעון ב 1 ואז בקו הנתון יש מעבר מ 0 ל 1 . מצב START חוזר משורטט בקו מקווקו ובו רואים שבזמן שקו השעון ב 1 יורד קו הנתון ל 0 .

העברת נתון בתקשורת I2C

שתי אפשרויות העברת נתונים קיימות בקו תקשורת I2C :

א. MASTER משרד וה- SLAVE קולט- אופן הכתיבה- Write Mode

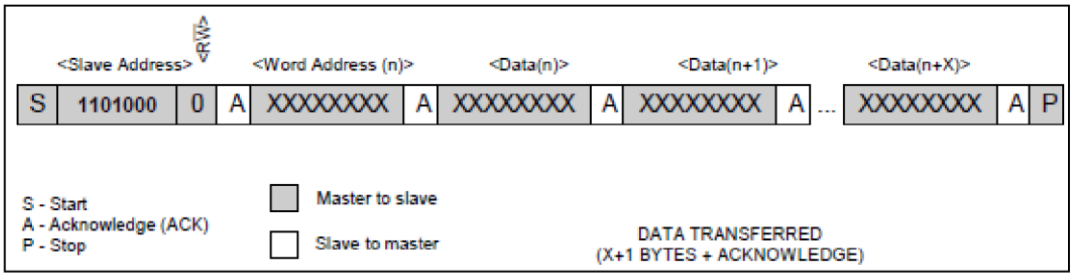
במקרה זה הביית הראשון המשודר על ידי ה MASTER הוא הכתובת של ה SLAVE)במקרה של רכיב DS1307 הכתובת היא 1101000X - D0H במקרה של כתיבה לרכיב או D1H אם קוראים מהרכיב(. לאחר מכן

יבואו מספר בתים של נתונים. ה SLAVE מחזיר ACKNOLEDGE בסיום כל ביית נתונים שקלט. הנתון מועבר עם ביט ה MSB ראשון !!

ב. byte משודר מה SLAVE אל ה MASTER - אופן קריאה - Read Mode

במקרה זה ה byte הראשון שנשלח הוא על ידי ה MASTER השולח את כתובת ה SLAVE שמחזיר מצדו את bit ה ACKNOWLEDGE . מכאן ה - SLAVE שולח מספר בתי נתונים. ה MASTER מחזיר ביט ACKNOLEDGE אחרי כל קליטת ביית נתון חוץ מהביית האחרון שהוא איננו מחזיר ACNOWLEDGE או אפשר להגיד שהוא מחזיר Not ACKNOWLEDGE .

אופן כתיבה – ה MASTER משדר אל אחד מה SLAVES



איור 7 - אופן כתיבת נתון מה- MASTER כשה- SLAVE הוא המקלט .

באיור 7 מתואר מצב שבו ה MASTER כותב אל ה SLAVE המשמש כמקלט. החלק הכהה שבאיור הוא מה ששולח ה MASTER. החלק הבהיר הוא מה ששולח המקלט – ה SLAVE.

ה- MASTER יוצר מצב START (מסומן ב S). לאחר מכן הוא שולח 7 ביטים של כתובת הרכיב - D0H במקרה של הרכיב DS1307 - והביט ה 8 הוא 0 המציין שהוא הכותב וה- SLAVE הוא המקלט. על ה SLAVE לענות ב(ACKNOLEDGE) מסומן ב- A. לאחר מכן ה- MASTER שולח ביית נוסף הטוען את מצביע (אוגר) הכתובות בתוך הרכיב. הנתון הבא נכתב לכתובת זו ומצביע הכתובות גדל אוטומטית ב 1. כל נתון נכתב בכתובת שבמצביע הכתובות ומצביע הכתובות מתקדם ב 1 . אחרי כל ביית שנקלט על ידי ה SLAVE הוא שולח אישור שקלט – ACKNOELEDGE . ה MASTER מסיים את התקשורת בעזרת מצב STOP)מופיע בצד ימין עם האות P.

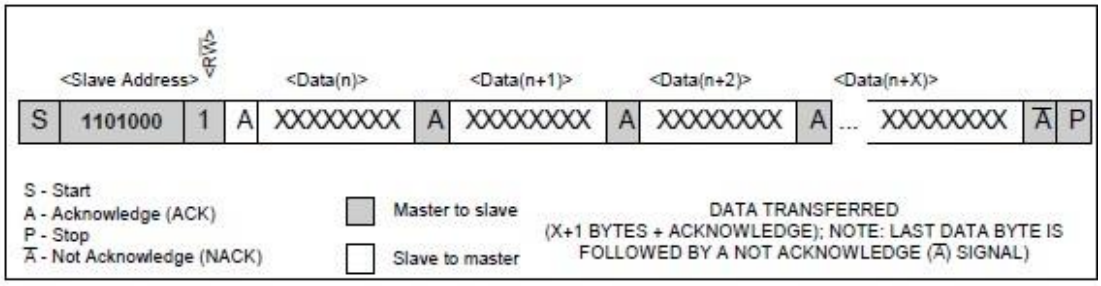
אופן קריאה - ה- SLAVE משדר אל ה- MASTER

גם מצב זה מתחיל תמיד במצב שבו ה- MASTER משדר אל ה- SLAVE אבל כאן הוא אומר שהוא רוצה לקרא ממנו. הבית הראשון שה MASTER משדר נקלט על ידי ה SLAVE כמו שתואר בפסקה הקודמת, כלומר ה MASTER ייצור מצב START , יישלח את 7 הביטים של הכתובת 10110001 אבל הביט השמיני יהיה 1 שבו הוא אומר שהוא רוצה לקרא. מכאן ה SLAVE משדר את הנתונים וה MASTER עונה עם ביט ACKNOLEDGE . בביית האחרון , כשה

MASTER רוצה בסיום התקשורת , הוא איננו מגיב בביט ה 9 ולא שולח ACKNOLEDGE (מסומן באיור ב Â)וגם שולח ביט עשירי של STOP .

הנתונים המשודרים מה SLAVE מתחילים מהכתובת האחרונה שבה נמצא מצביע הכתובות. כל נתון שה SLAVE שולח הוא מקדם את מצביע הכתובות לכתובת הבאה – אוטומטית.

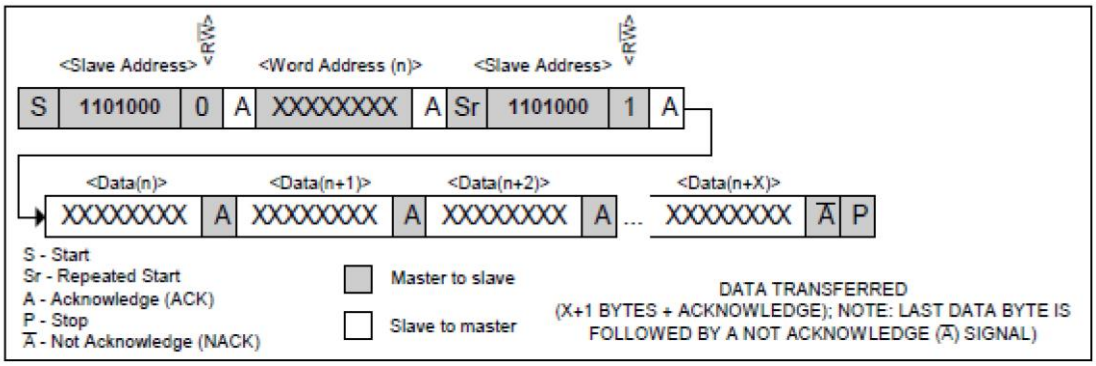
איור 8 מתאר מצב תקשורת זה .גם כאן הצבע הכהה הוא של ה MASTER והבהיר של ה SLAVE .



איור 8 - אופן קריאה – ה MASTER הוא המקלט

בדרך כלל תהליך התקשורת יהיה הבא : ה MASTER יכתוב 2 בתים אל הרכיב. בבית הראשון הוא אומר לרכיב שהוא פונה אליו לכתיבה. בבית השני הוא יציין את הכתובת הרצויה. מיד לאחר מכן יישלח STOP (או START חוזר) ואז יבצע תקשורת חדשה שבו הוא ייפנה לרכיב לקריאה מהכתובת ששלח אליו בפעולת הכתיבה.

איור 9 מתאר פעולת כתיבה וקריאה מהכתובת הרצויה. גם כאן הצבע הכהה הוא של ה MASTER והבהיר של ה SLAVE .



איור 9 – פעולה משולבת של כתיבה ל SLAVE כדי לציין כתובת רצויה וקריאה מה SLAVE

מהאיור רואים שה MASTER שלח 2 בתים אל ה SLAVE . מיד לאחר מכן יצר מצב START חוזר (מסומן ב Sr), שלח בית נוסף שבו יש את כתובת הרכיב 10110001 (הביט השמיני הוא של קריאה) ומרגע זה ה SLAVE משדר וה-MASTER מגיב ב ACKNOLEDGE . בסיום התקשורת ה MASTER לא נותן ACKNOLEDGE (מסומן ב Â) ולאחר מכן נותן מצב STOP .

תקשורת SPI

תקשורת טורית סינכרונית, (Serial Peripheral Interface) בין שני התקנים

Mater-Slave כאשר ה- Master מספק CLK למערכת,

וה -Slave שולח/מקבל מידע לפי שעון זה.

מערכת ה SPI - היא מערכת דופלקס מלאה (Full Duplex) כלומר, שליחת מידע וקבלת מידע (דרך 2 חוטים) בו זמנית.

על מנת להשתמש בתקשורת SPI יש צורך בארבעה קווים , 2 קווי בקרה ו- 2 קווי מידע/

MOSI- (Master Out Slave In) - מידע נשלח מה-Master אל ה-Slave.

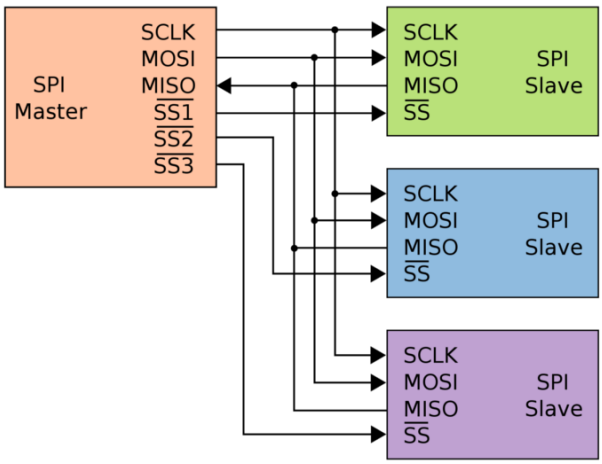
MISO- (Master In Slave Out) - מידע נשלח מה- Slave אל ה-Master .

SCLK- שעון טורי- קובע את תזמון העברת המידע בין ה- Master וה-Slave.

SS- (Slave Select)- קו זה פעיל בנמוך ומספק בקרה להתחלת פעולת התקשורת עם ה-Slave.

חיבור פיסי

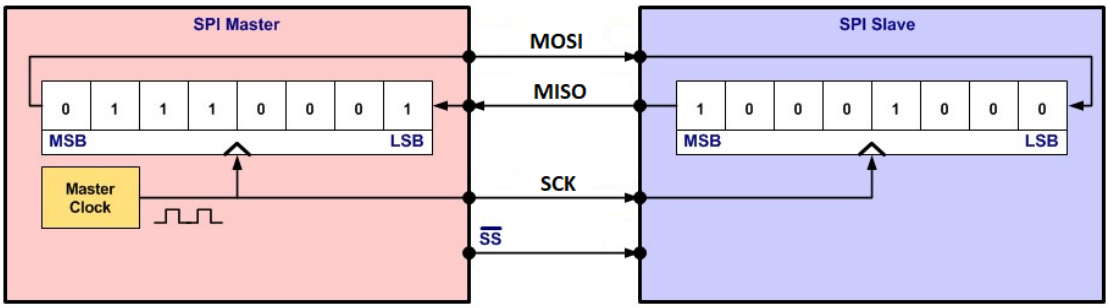
בתקשורת זו יש אפשרות לחבר מספר התקני Slave במקביל, כאשר ה- Master מוריד את הדק SS של אחד ה- Slave בלבד .



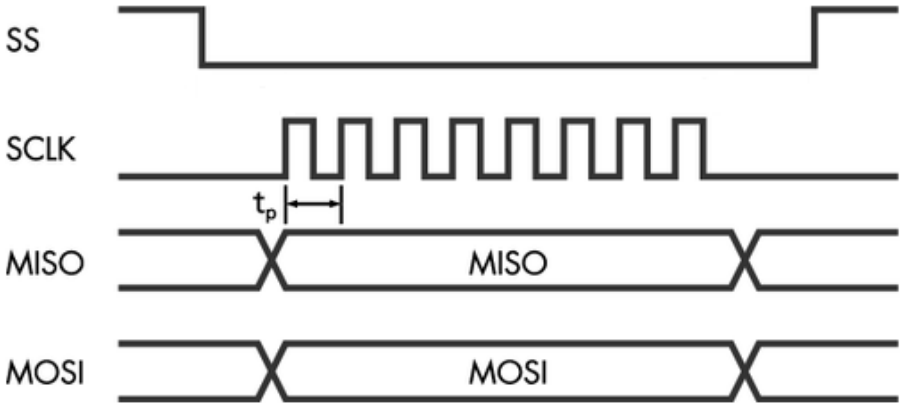
פרוטוקול תקשורת

המידע מועבר בו זמנית בשני הכיוונים בכל עליית שעון באמצעות אוגרי הזזה .

בדוגמה הבאה: לאחר 8 מחזורי שעון מידע מוחלף בין ה- Master ל-Slave.



שליחת מילת בקרה של 8 סיביות.



בתחילת התקשורת, הדק SS יורדת ל-0 . לאחר מכן מועבר המידע מה-Master דרך הדק MOSI. במקביל ה-Master יכול לקרוא את המידע מה- Slave דרך הדק MISO

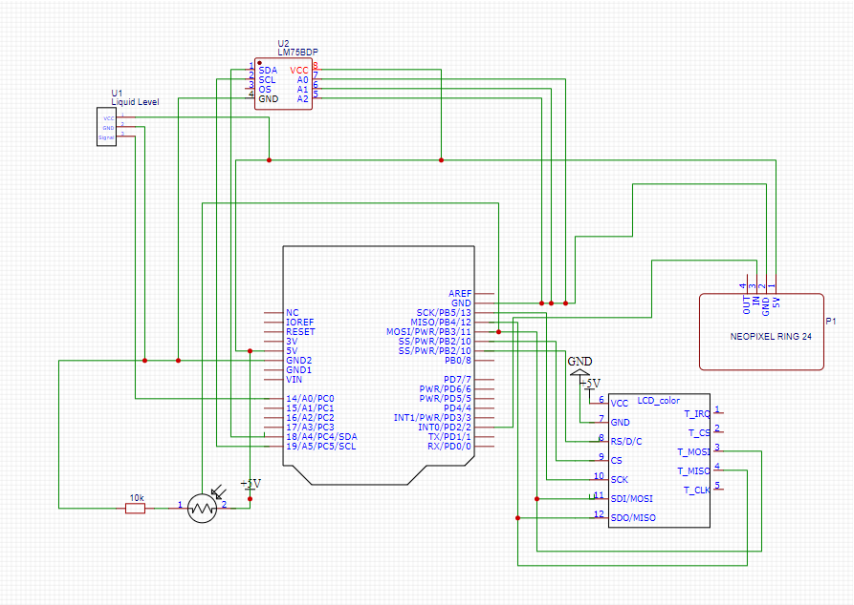
בסיום התקשורת הדק SS חוזרת ל-1 לוגי.

tp – זמן מחזור של השעון.

חיישן רמת מים (Water Level Sensor) בדרך כלל פועל במצב מתג (Switch) , הוא אינו משתמש בפרוטוקול תקשורת מסוים אלא מפיק אות דיגיטלי בהתאם לגובה המים.

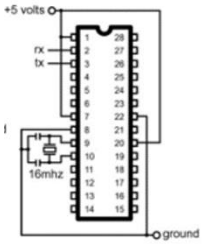
תמונה של הפרויקט

שרטוט חשמלי



פרוט על מרכיבי הפרויקט

- Arduino Uno בקר

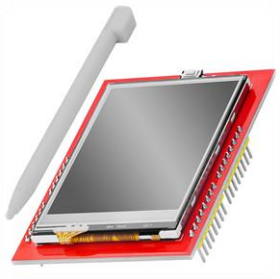


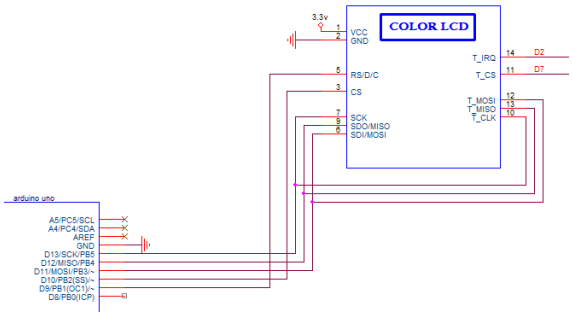
המאפיינים העיקריים של הרכיב Atmega328p הם:

- תדר שעון 16MHZ.
- מתח עבודה 5v (אספקת מתח לכרטיס 7v-12v).
- זרם בהדקי I/O עד 40mA.
- זיכרון תוכנית (flash) בגודל 32k.
- זיכרון נתונים (ram) בגודל 2k.

- 14 כניסות ויציאות דיגיטאליות.
- 6 יציאות PWM.
- 6 כניסות אנאלוגיות ברזולוציה של 10 סיביות.
- תקשורת טורית (spi, i2c , rs232).
- 2 פסיקות חיצוניות.

-מסך צבעוני





גרפי צבעוני, ברזולוציה של 240x320 פיקסלים RGB, כל פיקסל מקבל ערך של 16bit או 18bitהמרכיבים את הצבע.

שליחת המידע היא באמצעות תקשורת טורית SPI המורכב מ-4 הדקים: MISO, MOSI, CS , SCK (ניתן לחבר 2 מערכות : צג, מסך מגע).

ועוד שני רגלי בקרה: הדק RS לקביעת נתון או בקרה ו-RESET לאיפוס.

תפקיד הפינים:

Reset- מאתחל את מעבד התצוגה.

AD/RS- הדק הקובע אם נשלח מילת בקרה (כתובת פיקסל) RS=0

או קובע אם נשלח מידע של הצבע RS=1.

MOSI/SDI- קו כניסת נתונים טורי.

MISO/SDO- קו יציאת נתונים טורי.

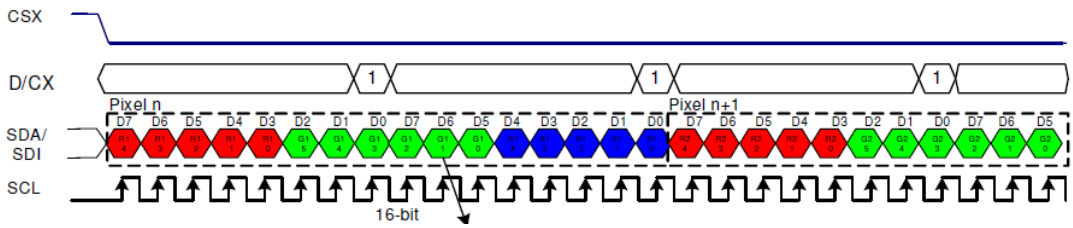
SCK- קו שעון ששולח המעבד החיצוני.

CS-הדק אפשור תקשורת. כאשר מתחילה תקשורת, ההדק יורד ל-0.

קביעת צבע לכל פיקסל

ניתן לשלוח מידע על צבע הפיקסל לפי 16bit ואז שולחים מידע של שני בתים המכילים מידע על צבע ירוק, אדום או כחול.

בתוך המילה המשודרת של 16Bit יש את כל הצבעים לפי התיאור הבא:



5 ביטים- מרכיב של צבע אדום.

6 ביטים- מרכיב של צבע ירוק.

5 ביטים- מרכיב של צבע כחול.

בעזרת שילוב של כל הצבעים ניתן לקבל 65K צבעים, כדי להגיע לכל פיקסל יש צורך לכתוב קודם את הכתובת ואחר כך את המידע.

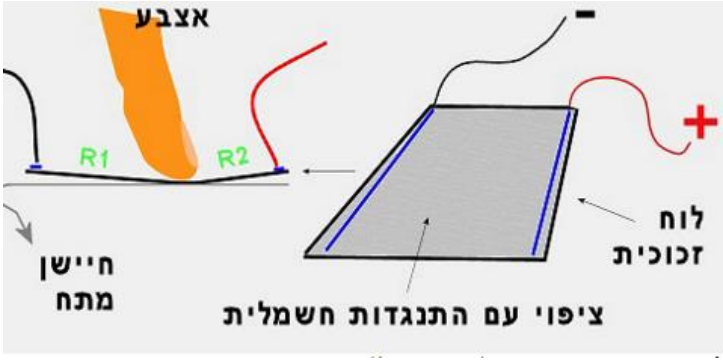
לפי התרשים רואים שבהתחלת התקשורת CS=0 ולאחר מכן משדורים 16 ביטים בצורה טורית.

כל עליית שעון מועבר ביט אחד לפי הפרוטוקול.

כדי לקבוע אם שידור הנתונים יהיה כתובת או מידע נשנה את המצב הלוגי של RS.

- מסך TOUCH

עקרון הפעולה



משטח מצופה התנגדות מכוסה בזכוכית דקה (כפי שמתואר באיור), בזמן נגיע במסך למעשה מתבצעת חלוקה של המסך לשני נגדים R1, R2 בציר Y. על מנת ליצור חלוקה גם בציר X, הכניסו משטח נוסף כך שמתבצעת חלוקה לשני נגדים גם בציר זה .

חלוקה לשני נגדים בכל נקודת מגע מאפשרת חלוקת מתח, המתח תמיד יהיה יחסי לנגיעה.

באמצעות ממיר ADC ברזולוציה של 12 סיביות דו ערוצי (Y,X)הנמצא על המסך אני ממירים את המתח למידע דיגיטלי הנשלח החוצה למעבד בתקשורת טורית SPI .

כעת נותר רק להתאים את התוצאות המתקבלות לכמות הפיקסלים על המסך.

הרגלים המסומנות ב- T הן האחראיות לתקשורת הטורית , 4 מהן בפרוטוקול SPI , T_IRQ יורד ל- '0' בזמן נגיעה על המסך.

למעשה באמצעות תקשורת זאת אנו קוראים את ערכי הממיר לצירים X,Y מהמסך.

הסבר פונקציות – LCD Touch

שם הפונקציה	תיאור
void begin()	אתחול התצוגה

void fillScreen(uint16_t color)	צובעת את כל המסך

פירוט: פונקציה מקבלת color – צבע המסך לפי 16 סיביות (5bit RED, 6bit Green, 5bit Blue) או שם באותיות גדולות של הצבע.	`#define BLACK 0x0000`
	`#define GRAY 0x8410`
	`#define BLUE 0x001F`
	`#define RED 0xF800`
	`#define GREEN 0x07E0`
	`#define CYAN 0x07FF`
	`#define MAGENTA 0xF81F`
	`#define PURPLE 0x7194`

חלק מהצבעים:	`#define ORANGE 0xFE00`
	`#define YELLOW 0xFFE0`
	`#define WHITE 0xFFFF`

	LcdTouch.fillScreen (RED); // דוגמה
	=
	0xF800

void setRotation(uint8_t m)	מסובבת את המסך ל- 4 כוונים פונקציה מקבלת – ערך 0 עד 3

void setTextSize(uint8_t fontSize)	קובעת את גודל הגופן פונקציה מקבלת –ערך של גודל פונט 1 ומעלה (ברירית מחדל 2)

void setCursor (uint16_t x,uint16_t y)	בחירת מיקום על המסך . פונקציה מקבלת – ערכי X, Y

void setTextColor(uint16_t color)	משנה את צבע הטקסט. פונקציה מקבלת color – צבע המסך לפי 16 סיביות

void setTextColor(uint16_t color, uint16_t background)	קובעת צבע הטקסט ורקע הטקסט (הרקע מוחק טקסט קודם) פונקציה מקבלת – צבע הטקסט וצבע הרקע של הטקסט. דוגמה setTextColor(BLACK,RED);

void print (int num)	פונקציה המדפיסה את ערכו המספרי של המשתנה. פונקציה מקבלת – משתנה בגודל 16 סיביות דוגמה המדפיסה את ערכו של print (x) ; x

void print (char *str)	פונקציה המדפיסה מחרוזת. פונקציה מקבלת –מחרוזת. דוגמה print ("Press Keyboard to return");

void println (int num)	פונקציה המדפיסה את ערכו המספרי של המשתנה ויורדת שורה. פונקציה מקבלת – משתנה בגודל 16 סיביות

void println (char *str)	פונקציה המדפיסה מחרוזת ויורדת שורה. פונקציה מקבלת – מחרוזת.

void print(uint16_t x, uint16_t y, char *str, uint16_t fontSize, uint16_t fColor)	תיאור – כתיבת הודעה על המסך עם קביעת המיקום, גודל הפונט, צבע ההודעה. פונקציה מקבלת –מיקום ההודעה x y , מחרוזת, גודל הפונט, צבע ההודעה. דוגמה print(40, 20, "hello", 2,RED);

void print(uint16_t x, uint16_t y, char *str, uint16_t fontSize, uint16_t fColor, uint16_t bColor)	כתיבת הודעה על המסך עם קביעת המיקום, גודל הפונט, צבע ורקע ההודעה. פונקציה מקבלת –מיקום ההודעה x y , מחרוזת, גודל הפונט, צבע ההודעה, צבע הרקע. דוגמה print(40, 20, "hello", 2,RED,BLACK);

void printheb(char *str)	פונקציה המקבלת מחרוזת תווים להצגה על המסך בעברית . פונקציה מקבלת –מחרוזת בעברית דוגמה printheb("שלום");

void printhebln(char *str)	פונקציה המדפיסה הודעה בעברית על המסך ויורדת שורה. פונקציה מקבלת –מחרוזת בעברית

void printheb(uint16_t x, uint16_t y, char *str, uint16_t fontSize, uint16_t fColor)	כתיבת הודעה בעברית על המסך עם קביעת המיקום, גודל הפונט, צבע ההודעה. פונקציה מקבלת –מיקום ההודעה x y , מחרוזת להצגה, גודל הפונט, צבע ההודעה. דוגמה printheb(10,20,"שלום",2,RED);

void printheb(uint16_t x, uint16_t y, char *str, uint16_t fontSize, uint16_t fColor, uint16_t bColor)	תיאור – כתיבת הודעה בעברית על המסך עם קביעת המיקום, גודל הפונט, צבע ההודעה והרקע. פונקציה מקבלת – מיקום ההודעה x y , מחרוזת להצגה, גודל הפונט, צבע ההודעה, צבע הרקע. דוגמה printheb(10,20,"שלום" ,2,RED,BLACK);

void drawPixel (int16_t x,int16_t y, uint16_t color)	תיאור – צובעת פיקסל אחד. פונקציה מקבלת –מיקום הפיקסל x y , צבע הפיקסל. דוגמה drawPixel (100 ,150 , GREEN);

void drawHLine (int16_t x,int16_t y, int16_t w, uint16_t color)	תיאור – שרטוט קו אופקי פונקציה מקבלת –מיקום תחילת הקו x y , אורך הקו, צבע הקו. דוגמה drawHLine (20,40, 100, ORANG) ;

void drawVLine (int16_t x,int16_t y, int16_t h, uint16_t color)	תיאור – שרטוט קו אנכי פונקציה מקבלת – מיקום תחילת הקו x y , גובה הקו, צבע הקו. דוגמה drawVLine (30,50, 100, BLUE) ;

void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color)	תיאור – ציור קו פונקציה מקבלת –ערכי התחלת הקו x0 y0 , סוף הקו x1 y1 , צבע הקו.

void drawRect (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)	תיאור – שרטוט מסגרת מלבן בצבע. פונקציה מקבלת –ערכי x y של צלע עליונה משמאל של המלבן, אורך ורוחב , צבע המסגרת. דוגמה drawLine(10, 10, 100, 100, BLACK);

void drawRoundRect(int16_t x, int16_t y, int16_t w,int16_t h, int16_t r, uint16_t color)	תיאור – שרטוט מסגרת מלבן בצבע, עם עיגול הפינות תיאור – שרטוט מסגרת משולש. פונקציה מקבלת – ערכי x y של צלעות המשולש, צבע מסגרת המשולש. דוגמה drawTriangle(10, 100, 100, 100,20 ,20, RED);

void drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)	תיאור – שרטוט מסגרת עיגול. פונקציה מקבלת – מקבלת מיקום מרכז המעגל, רדיוס המעגל, צבע. דוגמה drawCircle(100, 100, 20 , BLACK);

void fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)	תיאור – שרטוט מלבן מלא בצבע נבחר. פונקציה מקבלת –ערכי x y של צלע עליונה משמאל של המלבן, אורך ורוחב , צבע המלבן. דוגמה fillRect (10, 10, 100, 100, BLACK) ;

void fillRoundRect(int16_t x, int16_t y, int16_t w, int16_t h, int16_t r, uint16_t color)	תיאור – – שרטוט מלבן מלא בצבע נבחר, עם עיגול הפינות פונקציה מקבלת –ערכי x y של צלע עליונה משמאל של המלבן, אורך ורוחב, רמת העיגול בפינות , צבע . דוגמה fillRoundRect(10, 10, 50,100, 5, RED);

void fillTriangle (int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2, uint16_t color)	תיאור – שרטוט משולש מלא בצבע נבחר. פונקציה מקבלת – ערכי x y של צלעות המשולש, צבע . דוגמה drawTriangle(10, 100, 100, 100,20 ,20, RED)

void fillCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)	תיאור – שרטוט עיגול מלא בצבע נבחר. פונקציה מקבלת – –מקבלת מיקום מרכז המעגל, רדיוס המעגל, צבע.

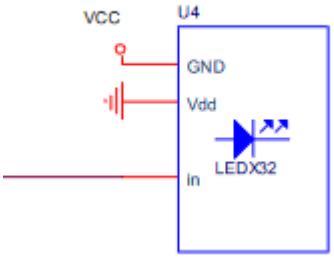
void drawButton(int8_t NumButton, int16_t x, int16_t y, int16_t w, int16_t h, int16_t r, uint16_t Color, uint16_t textcolor, char *label, int8_t textsize)	תיאור – ציור פקד במסך עם טקסט. פונקציה מקבלת – מספר הפקד, מיקום x,y גובה –h ,רוחב – w, עיגול הפינה – r, צבע הפקד, צבע הטקסט, טקסט, גודל הגופן דוגמה drawButton(1, 20, 20, ; 90, 50, 10, RED, WHITE, "EXAM1", 2)

int8_t ButtonTouch(int16_t x, int16_t y)	תיאור – בודקת מספר הפקד שנלחץ לפי מיקום. פונקציה מקבלת – מיקום הלחיצה x, y פונקציה מחזירה – מספר הפקד דוגמה int ButtonNum = LcdTouch.ButtonTouch(LcdTouch.xTouch, LcdTouch.yTouch);

bool touched()	תיאור – בודקת אם יש לחיצה פונקציה מחזירה – לחוץ או לא

void readTouch()	תיאור – קריאת מיקום הלחיצה והכנסתם למשתני המחלקה: xTouch, yTouch
	דוגמה
	LcdTouch.readTouch();
	x = LcdTouch.xTouch;
	y = LcdTouch.yTouch;

-בר לדים Neo Pixel



בר הלדים מורכב ממספר לדים מסוג RGB, לכל לד שלושה צבעים שניתן להדליק כל צבע בנפרד או שילוב שלהם. ניתן לקבע באיזו עוצמה נרצה כל צבע כאשר 0 היא העוצמה הנמוכה ביותר (הצבע לא יידלק) ו 255 היא העוצמה הגבוהה ביותר.

לבר הלדים מספר חיבורים:

GND-נחבר לאדמה

VCC- נחבר ל 5V

Di- נחבר אל אחד הפורטים הדיגיטליים של הארדואינו, בדוגמה זו אל PIN 6.

Do- משמש על מנת לשרשר בר לדים נוסף. (בו לא נשתמש בפרויקט הנוכחי).

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(10, PIN, NEO_GRB + NEO_KHZ800);
```

10 - מספר הלדים אותם אנו רוצים להדליק

PIN- הפורט אליו ה Vin מחובר.

NEO_GRB + NEO_KHZ800- סוג בר הלדים (ירוק, אדום, כחול)+קצב הסיביות (800kHz)

בפונקציה setup: הכנת PIN 6 להוצאת מידע. strip.begin();

אתחול כל הפיקסלים למצב כבוי. strip.show();

הפונקציה colorWipe מדליקה את הלדים אחד אחרי השני בצבע זהה,

היא מקבלת שני פרמטרים:

C הקובע את עוצמת כל אחד מהצבעים ו wait הקובע את ההשהיה בין הדלקות הלדים.

דוגמה לשליחה לפו' colorWipe:

```
colorWipe(strip.Color(255, 0, 0), 50); // Red
```

תוכנית המפעילה את מעגל הלדים ומריצה בו צבעים:

```
#include <Adafruit_NeoPixel.h>

#define PIN 6

// Parameter 1 = number of pixels in strip
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(16, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
  // Send a theater pixel chase in...
  theaterChase(strip.Color(127, 127, 127), 50); // White
  theaterChase(strip.Color(127,   0,   0), 50); // Red
  theaterChase(strip.Color(  0,   0, 127), 50); // Blue

  rainbow(20);
  rainbowCycle(20);
  theaterChaseRainbow(50);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
  }
}
```

```
void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

//Theatre-style crawling lights.
void theaterChase(uint32_t c, uint8_t wait) {
  for (int j=0; j<10; j++) {  //do 10 cycles of chasing
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, c);    //turn every third pixel on
      }
      strip.show();

      delay(wait);

      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
```

בר הלדים העגול נבחר על ידי כדי לייצג להבות של הכיריים שתחת הסיר.

חיישן רמת מים:

חיישן מפלס המים Arduino הוא מכשיר המשמש למדידת מפלס המים בכלי או במיכל.

חיישן זה בעל שימושים רבים, כגון גילוי רמת משקעים, רמת הצפה, דליפות נוזל, מערכות אוטומציה ביתית, מערכות השקייה אוטומטיות, ניקוי תוצרים, מדידות איכות המים ועוד.

החיישן מורכב בעיקרו משלושה חלקים: מחבר אלקטרוני, נגד 1 M ומספר קווים מקבילים של חוטים מוליכים חשופים. חוטים מוליכים חשופים מחוברים לאדמה וביניהם שלובים דרכי החישה. נגד-UP PULL בערך של M1 גורם לחיישן לערך גבוה. כיוון שמים מוליכים חשמל בקלות, נפילת טיפת מים מקצרת את החיישן למוליכי האדמה וגורמת לו לערך נמוך.

לחיישן זה יש 3 רגליים:

רגל אחת של החיישן (-) הולכת לאדמה (GND-),

רגל שנייה של החיישן (+) הולכת ל- V5,

רגל שלישית (S) מספקת לנו את התוצאה, ולכן נחבר אותה לפין הכניסה האנלוגי A0-A6 או לפין הכניסה הדיגיטלי D1-D13 בארדואינו.

הפלט מתקבל על-ידי הכנסת החיישן לנוזל ומזהה את גובה המים בהתבסס על מוליכות או קיבול.

סוג הפלט הינו אנלוגי המספק טווח רציף של ערכים התואם למפלס המים.

שטח אזור חישה הינו 40 מ"מ – 16 מ"מ.

החיישן פולט ערכים בין 400 ל-800 בערך בהתאם למפלס המים.

פרוטוקולים : בדרך כלל פועל במצב מתג (Switch) , הוא אינו משתמש בפרוטוקול תקשורת מסוים אלא מפיק אות דיגיטלי בהתאם לגובה המים.

דוגמא לקוד שמשתמש בחיישן זה והסבר לפקודות

```
// Sensor pins
#define sensorPower 7
#define sensorPin A0

// Value for storing water level //ערך לאגירת מפלס המים
int val = 0;

void setup() {
        // Set D7 as an OUTPUT //D7 -פלט ב
        pinMode(sensorPower, OUTPUT);

        // Set to LOW so no power flows through the sensor
        //הגדר ל LOW כך שלא יזרום כוח דרך החיישן
```

```
        digitalWrite(sensorPower, LOW);

        Serial.begin(9600);
}

void loop() {

        //get the reading from the function below and print it
        //קבל את הקריאה מהפונקציה למטה והדפיס אותה

        int level = readSensor();

        Serial.print("Water level: ");
        Serial.println(level);

        delay(1000);
}


//This is a function used to get the reading
//זוהי פונקציה המשמשת לקבלת הקלט
int readSensor() {
        digitalWrite(sensorPower, HIGH);// Turn the sensor ON
        delay(10);              // wait 10 milliseconds
      val = analogRead(sensorPin);    // Read the analog value form
sensor
      digitalWrite(sensorPower, LOW); // Turn the sensor OFF
      return val;                     // send current reading
}
```

חיישן טמפרטורה LM75

ספריהWire (לשימוש בחיישן LM75)

פונקציהWire.begin

זוהי פונקציה אשר באמצעותה מאתחלים את יחידת.I2C\TWI

פונקציהWire.beginTransmissionaddress()

זוהי פונקציה אשר באמצעותה מתחילים את התקשורת בין המיקרו-בקר עם רכיב ה.Slave -

פונקציהWire.endTransmission()

באמצעות פונקציה זו עוצרים את התקשורת בין המיקרו-בקר) המשמש כ (Master -ורכיב ה-Slave.

פונקציהWrite(Parameters)

באמצעות פונקציה זו כותבים מהמיקרו-בקר אל לרכיב ה.Slave -

דוגמה לשליחת נתון

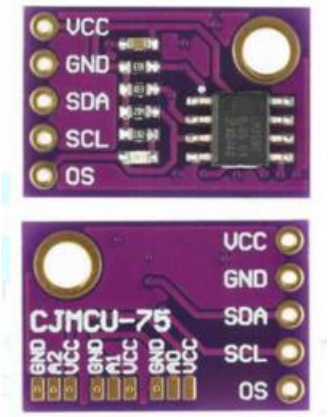
להלן דוגמה לשליחת הערך של המשתנה לרכיב Slave בכתובת 8. ערך המשתנה גדל

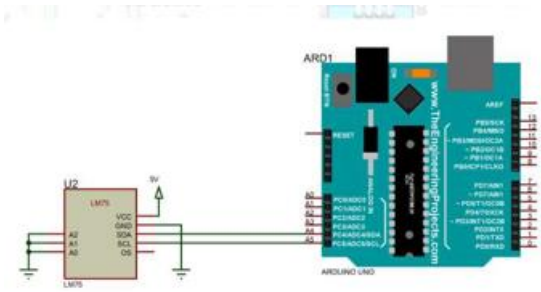
```
#include <Wire.h> // library i2c
void setup() {
Wire.begin();
// join i2c bus
}
byte val
```

```
= 0;
ב-1 כל שנייה.
void loop() {
Wire.beginTransmission(8); //transmit to device 8
Wire.write(val);
Wire.endTransmission();
// sends data byte
// stop transmitting
val++; // increment value
delay(1000);
```

פונקציהWire.requestFrom
באמצעות פונקציה זו ניתן לקרוא נתון או נתונים מרכיב ה Slave -והיא מקבלת שני פרמטרים:
הפרמטר הראשון מציין את כתובת רכיב ה Slave -שעמו רוצים לתקשר. והפרמטר השני קובע
את מספר הנתונים בגודל Byte שרוצים לקרוא.
פונקציהWire.available
מחזירה את מספר הנתונים שנקראו מרכיב הSlave -
פונקציה Wire.read
באמצעות פונקציה זו ניתן לקרוא נתון בגודל Byte מרכיב ה.Slave -
דוגמה לקריאת נתון
להלן תוכנית לקריאת נתון מתקשורת i20 מרכיב Slave בכתובת 8 והצגתו על גבי מסך

```
#include <Wire.h>
void setup() {
Wire.begin(8);
Serial.begin(9600);
התקשורת הטורית.
// slave device #8
// start serial for output
void loop() {
while (Wire.available())
{
char c = Wire.read(); // receive a byte as character
Serial.print(c);
// print the character
}
}
delay(500);
```





המודול הנ"ל כולל את חיישן הטמפרטורה . LM75 נגדי Pullup המחוברים להדקי התקשורת קבל סינון ופסים להלחמה לקביעת כתובת הרכיב.
LM75הוא חיישן טמפרטורה דיגיטלי הפועל בפרוטוקול C בתדר עבודה של עד KHz400 וכולל מערכת הממירה את הטמפרטורה ממתח אנלוגי למידע דיגיטלי בגודל 11 סיביות עם סימן.
רזולוציית רכיב הטמפרטורה היא °0.125 ,Cותחום מדידת הטמפרטורה נע בין °55 -Cל- +125°C .
הדקי הרכיב
– VCCמתח אספקה בין 2.8 Vל- V5.5
- GNDאדמה
– SDAהדק מידע ספרתי דו כיווני של פרוטוקול התקשורת הטורית.i2c
– SCLהדק כניסת שעון של פרוטוקול התקשורת הטורית.i2c
– OSהדק מוצא open drain הפעילה כאשר הטמפרטורה עולה מעל סף שנקבע.
A2, A1, A0
-
קווי כתובת של הרכיב המאפשר חיבור של עד 8 רכיבים במקביל לקוי התקשורת.

פרוטוקל התקשורת:

I2C

תוכנית דוגמה לקריאת הטמפרטורה מרכיבLM75
הערה: תוכנית עבור ערכי טמפרטורה חיובים בלבד, כדי לקבל את הטמפרטורה ללא נקודה עשרונית נגדיר את TEMP כint -

```
#include <Wire.h>
ax +0x48;
const int LM75_addr=0x48; //1001000 12c address of the LM75
float TEMP;
void setup() {
Wire.begin();
820
}
Wire.beginTransmission (LM75_addr);
Wire.write(0x00); // Temperature register (Temp)
Wire.endTransmission (true);
Serial.begin(9600);
void loop() {
Wire.beginTransmission (LM75_addr);
Wire.requestFrom (LM75_addr, 2, true); // request 2 registers
TEMP ((Wire.read() << 8 Wire.read()) >> 5) 0.125;
Serial.print (TEMP);
Serial.println(" OC");
delay(1000);
}
```

תיעוד הפרויקט

שלב א'- מציאתי רעיון לפרויקט וחקרתי את החיישנים שלו.

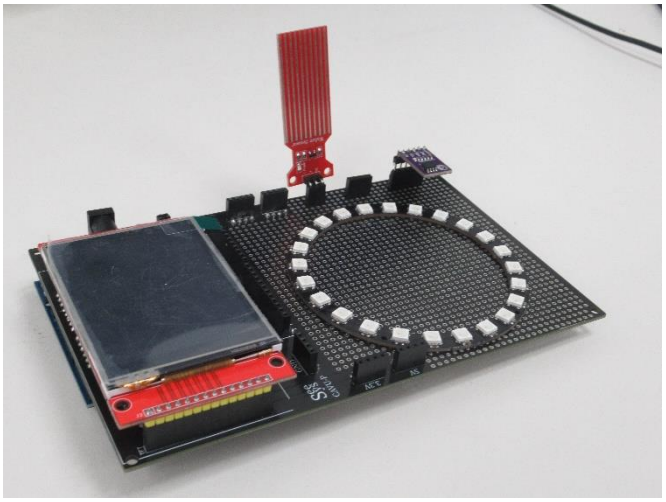
שלב ב' - חיבור הרכיבים לכרטיס ה ARDUINO -התאמתי תושבות לרכיבים שבהם אני משתמשת: חיישן מפלס מים, חיישן טמפרטורה , מסך מגע, בר לדים .הלחמתח אותם לכרטיס, והרכבתי עליהם את הרכיבים.

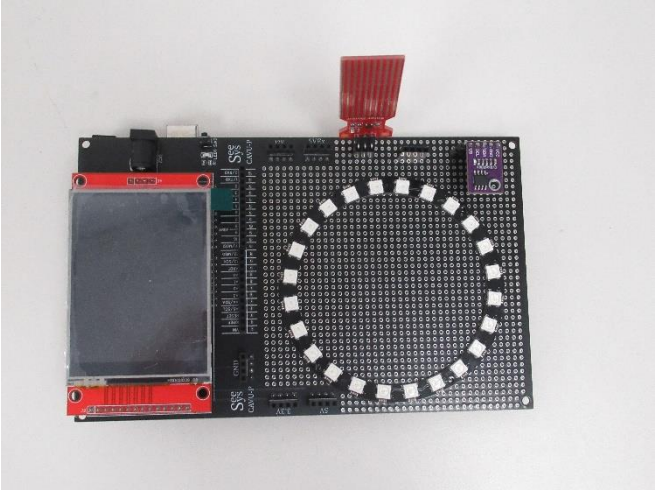
לאחר מכן, הייתי צריכה להלחים את הרכיבים בעזרת בדיל ומלחם על התושבות שעל גבי כרטיס הארדואינו .

שלב ב' / חיווט על ידי ה wrap wire -חיווטתי את כל הרכיבים לרגליים המתאימות להם בכרטיס הארדואינו, זאת על פי השרטוט החשמלי .באופן כללי, השתמשתי בצבעים המקובלים לחיווט: חוט אדום למקור המתח(vcc) חוט שחור לאדמה (GND) את שאר הצבעים בחרתי שרירותית .גם בשלב זה השתדלתי שהמראה יהיה אסתטי והחוטים מתוחים דיים, על מנת שאוכל לאחר מכן, במקרה של תקלה, לוודא שהחוטים אכן מחוברים לרגליים הנכונות . בשלב הבא התחלתי בחיבור הרכיבים עצמם כדלהלן :א. חיישן טמפרטורה :רגל אחת ל VCC רגל שניה ל GND רגל שלישית ל ש0 A ב. חיישן מפלס מים :חיברתי רגל אחת ל VCC רגל שניה ל GND רגל שלישית 1 A. מסך :מחובר ישירות לארדואינו ד. בר לדים :רגל אחת ל VCC רגל שניה ל GND רגל שלישית מחוברת לרגל דיגיטלית מס 2

שלב ג' - תוכנה לאחר שבניתי תרשים זרימה של התוכנה, התחלתי לעבוד על פיו ולכתוב את התוכנית עצמה. תחילה כתבתי בתכנת הארדואינו תוכנית מתאימה עבור כל רכיב ,לאחר מכן וידאנו כי הרכיב פועל כראוי ולבסוף חיברנו את כל קטעי התכנית לתכנית אחת על פי סדר פעולות הפרויקט.

תמונות:





תקלות:

- הכרטיס ארדואינו באחת מהפעמים שישבתי על הפרויקט התחמם בצורה לא רגילה ורק לאחת מספר ניסיונות ובדיקות גיליתי שחיברתי אותו לתושבת הפוך והרגלים לא עמדו במקומות המתאימים להם. כמובן שהפכתי אותו והכל בא על מקומו בשלום.
- החיישן טמפרטורה לא נתן לי תוצאה נכונה. ניסיתי להחליף חיישנים שמא החיישן תקול, ניסיתי קטעי קודים שונים ולאחר מספר ימים התברר לי שאני צריכה להלחים את פסי ה A0, A1, A2 שמצויים על החיישן לאדמה על ידי הלחמה. הלחמתי את החיישן הוא החל לפעול בלי תקלות.

- כאשר העלתי את המונה שעל מסך המגע מעל 10 או מעל 100 כאשר לחצתי להורדת המונה למספר ספרות קטן מזה נוספה הסיפרה 0 מצד ימין של המספר ושינה את ערכו. לאחר בדיקות והרצות חוזרות ונשנות מצאתי שהערך המספרי של המשתנה אינו שונה מהנחוץ אך הערך שמוצג על המסך מעוות והספרה הנוספת הינה בתצוגה בלבד ואינה משפיעה על הקוד ולכן הוספתי בדיקה האם הערך שווה ל 10 או ל100 ומשתמש לוחץ על הפחתת המונה אז תציג את הערך שהתקבל לי על המסך מתחילה.

תרשים זרימה


התחלה
בחר טמפרטורה
בחר את משך הזמן לבישול
לחץ על הכפתור להתחלת הבישול
הדלקת הלהבות תוך ניתור הטמפרטורה גובה המים, והזמן שעבר
כיבוי הלהבות
סיום

קוד תכנות

```
using namespace std;

#include "SPI.h"
#include "TFT9341.h"
#include "touch.h"

unsigned int x,y;
unsigned int counter=0;
int temp=100;

void setup  } ()
  lcd.begin;()
  lcdtouch.begin ;()
  lcdtouch.InitTypeTouch(2);//0,1,2
  lcd.setRotation(0);//0,1,2,3
  lcd.clrscr;()
  lcd.fillRoundRect (80,160,30, 40, 10 ,BLACK);
  lcd.fillRoundRect (220,160,30, 40, 10 ,BLACK);


'//  "'•'¨'"' ×'"//₪''ʔ
  lcd.setColor(WHITE);
  lcd.setFont)2;(
  lcd.gotoxy)90,170;(
  lcd.print  ;("-")



'//  "' ×'''¢'"//₪''ʔ
  lcd.setColor(WHITE);
  lcd.setFont)3;(
  lcd.gotoxy)230,170;(
  lcd.print;("+")


'//  "'ʔ¨'₪''ʔ•'¨'" '"' '\'—//×'¨'
  lcd.setColor(BLACK);
  lcd.setFont)3;(
  lcd.gotoxy)150,170;(
  lcd.print(counter);

'//  ›'•//×'¨'×'
  lcd.gotoxy)80,115  ;(
  lcd.print("Temperature");

//  lcd.gotoxy)70,137;(
 // lcd.print;("-----------")
```

```
//{  End of setup function

void loop} ()

‘§‘‘//  ‘’ ×‘‘ ‘”‘‘—‘™‘¦‘” ‘ ‘¢‘”//‘¡‘‘
  while(digitalRead(2)==1);
  lcdtouch.readxy;()
  x = lcdtouch.readx;()
  y = lcdtouch.ready;()

‘?§‘‘//  ™//‘
  if(x > 50 && x<130 && y> 160 && y< 200)}
    if(counter==0)
          counter=0;
    else}
      counter ;--
‘//      ‘’›‘“‘™ ‘ ‘‘©‘™‘¦‘™‘’ ‘ ×‘‘”‘¨‘₪‘¡‘‘™ ¡‘₪‘‘ ‘¢‘ ‘
‘‘‘¦‘“ ‘™‘‘™‘ ‘‘ ‘™ ‘‘›‘¡‘” ‘‘•‘ ‘×‘•‘¦‘‘™‘’‘” ‘‘—‘“//©‘
      if(counter == 99 || counter == 9)}
        lcd.gotoxy)150,170;(
        lcd.print;("   ")
        lcd.gotoxy)150,170;(
        lcd.print(counter) ;
        Serial.println(counter);
 {

{
    Serial.println(counter);

{


‘‘//  ’‘“‘™//‘
  if(x > 200 && x<280 && y> 160 && y< 200)}
    if(counter==255)
       counter=255;
     else
       counter ;++
     Serial.println(counter);
{

   if(temp != counter)}
     lcd.gotoxy)150,80;(
     lcd.print;("         ")
     lcd.gotoxy)150,170;(
     lcd.print(counter) ;
      analogWrite(5,counter);
      temp=counter ;
  {
{

using namespace std;

#include "SPI.h"
#include "TFT9341.h"
```

```
#include "touch.h"

unsigned int x,y;
unsigned int counter=0;
int temp=100;

void setup  } ()
  lcd.begin;()
  lcdtouch.begin ;()
  lcdtouch.InitTypeTouch(2);//0,1,2
  lcd.setRotation(0);//0,1,2,3
  lcd.clrscr;()
  lcd.fillRoundRect (80,160,30, 40, 10 ,BLACK);
  lcd.fillRoundRect (220,160,30, 40, 10 ,BLACK);


'//  ”'•'¨'“' ×'”//₪''˚
  lcd.setColor(WHITE);
  lcd.setFont)2;(
  lcd.gotoxy)90,170;(
  lcd.print  ;("-")



'//  ”' ×'''¢'”//₪''˚
  lcd.setColor(WHITE);
  lcd.setFont)3;(
  lcd.gotoxy)230,170;(
  lcd.print;("+")


'//  ”'˚¨'₪''˚•'¨'” '”' '`'—//×'¨'
  lcd.setColor(BLACK);
  lcd.setFont)3;(
  lcd.gotoxy)150,170;(
  lcd.print(counter);

'//  ›'•//×'¨'×'
  lcd.gotoxy)80,115  ;(
  lcd.print("Temperature");

//  lcd.gotoxy)70,137;(
 // lcd.print;("-----------")

//{  End of setup function

void loop} ()

'§''//  `' ×'' '”''—'™'¦'” ' '¢'”//''¡''
  while(digitalRead(2)==1);
  lcdtouch.readxy;()
  x = lcdtouch.readx;()
  y = lcdtouch.ready;()

'˚§''//  ™//'
  if(x > 50 && x<130 && y> 160 && y< 200)}
```

```
    if(counter==0)
          counter=0;
    else}
      counter ;--
،//       ‘،›،“،™ ، ،،©،™،¦،™،، ، ×،،”،¨،₪،¡،،™ ¡،₪،، ،¢، ،
،‘،¦،“ ،™،،™، ،، ،™ ،،›،¡،” ،،•، ،×،•،¦،،™،،،” ،،—،“//©،
      if(counter == 99 || counter == 9)}
        lcd.gotoxy)150,170;(
        lcd.print;("   ")
        lcd.gotoxy)150,170;(
        lcd.print(counter) ;
        Serial.println(counter);
 {

{
    Serial.println(counter);

{


،،//  ،،“،™//،
  if(x > 200 && x<280 && y> 160 && y< 200)}
    if(counter==255)
       counter=255;
     else
       counter ;++
     Serial.println(counter);
{

   if(temp != counter)}
     lcd.gotoxy)150,80;(
     lcd.print;("         ")
     lcd.gotoxy)150,170;(
     lcd.print(counter) ;
      analogWrite(5,counter);
      temp=counter ;
  {
{


int waterSensorPin = A0 ;

void setup} ()

  Serial.begin)9600;(
{

void loop} ()


  int sensorValue = analogRead(waterSensorPin);

    if(sensorValue > 0)}
      Serial.println(sensorValue);
{
    else
```

```
}
      Serial.println("no water");
{


{
```