# Hackathon Project

## Inspiratie

Dit project is gebaseerd op het arcadespel ‘Stacker’, of het mobiele spel 'Stack' (3D adaptatie), of iets soortgelijks.

<https://en.wikipedia.org/wiki/Stacker_(arcade_game)>

<https://play.google.com/store/apps/details?id=com.ketchapp.stack&hl=en&gl=US>

## Doel

De blokjes zo hoog mogelijk opstapelen terwijl ze (van links en rechts) op het scherm komen schuiven. Als een blok niet direct bovenop het vorige wordt geplaatst, wordt het overhangende gedeelte afgesneden, waardoor de stapel steeds korter wordt.  
Een simpel concept, maar effectief om de speler zijn timing en precisie uit te dagen.

* Een **8x8 LED matrix** zal het display zijn. De balkjes zullen 1 rij hoog zijn met het eerste balkje 5 breed. Telkens op de **drukknop** wordt gedrukt, zal een nieuw blokje zich stapelen op het vorige. Een bredere uitleg over de werking vindt u [hier](#_j0vzokbqaxb).
* De **DF Player Mini** zal een sound effect afspelen telkens een nieuw blokje succesvol geplaatst wordt.
* De **Infrarood afstandsbediening/receiver** kan het systeem aan/uit zetten, het volume controleren, en het spel resetten.
* De prijs bij het winnen zijn snoepjes in een doos, met een **servo** die het deksel sluit. Deze doos staat omgekeerd, en indien gewonnen wordt gaat de servo het deksel opendoen en vallen de snoepjes eruit. Een **ultrasone sensor** wordt boven in de snoepjesdoos geplaatst om te kijken of er nog prijzen in zitten. Een **led**-indicator geeft dit aan.

## Hardware

* Arduino MEGA
* 8x8 LED Matrix
* Servo-motor
* Infrarood afstandsbediening met reciever
* DF player MINI met SD-kaart
* LED
* Ultrasone Sensor
* Doos met snoepjes
* Drukknop

## Hoe het spel werkt

### **Basis Instelling:**

* De LED-matrix heeft 8 rijen en 8 kolommen, vergelijkbaar met het speelveld van 'Stacker'.
* De drukknop wordt gebruikt om de beweging van de blokken te stoppen.

### **Beweging van Blokken:**

* De blokken bewegen van links naar rechts en terug binnen een rij (*display\_row[row]*).
* De beweging wordt gerealiseerd door de bits in *display\_row[row]* te verschuiven:
  + **Links Verschuiving:** Verplaatst de blokken naar links.
  + **Rechts Verschuiving:** Verplaatst de blokken naar rechts.
* De *drawScreen()* functie ververst de LED-matrix continu om deze beweging weer te geven.

### **Stoppen van Blokken:**

* Wanneer de drukknop wordt ingedrukt (*buttonPressed()* functie), wordt de huidige rij vastgezet en gecombineerd met de vorige rij (*row-1*).
* Alleen de overlappende blokken worden behouden in de nieuwe rij. Dit simuleert het mechanisme van blokken die vallen als ze niet goed uitgelijnd zijn.

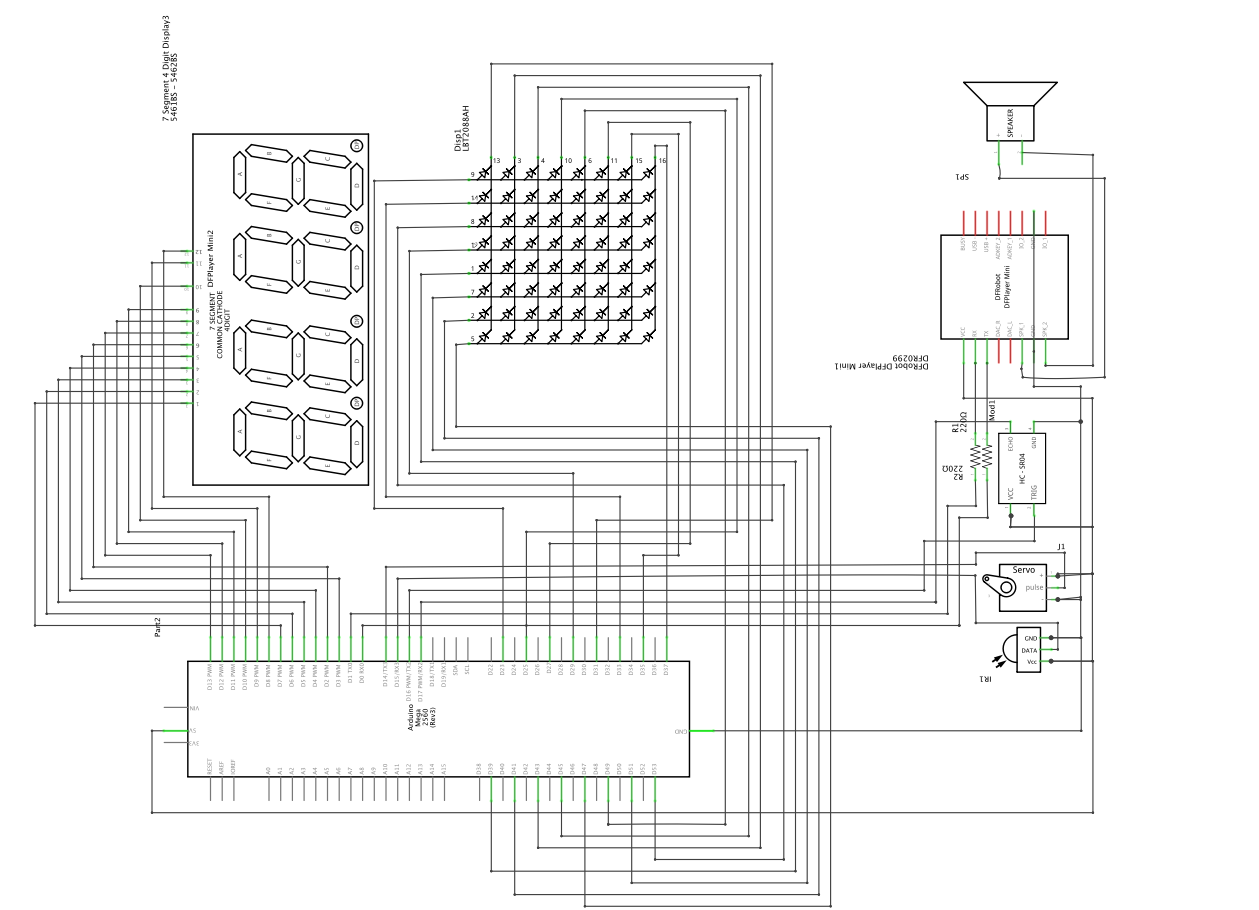
### **Grootte Berekening:**

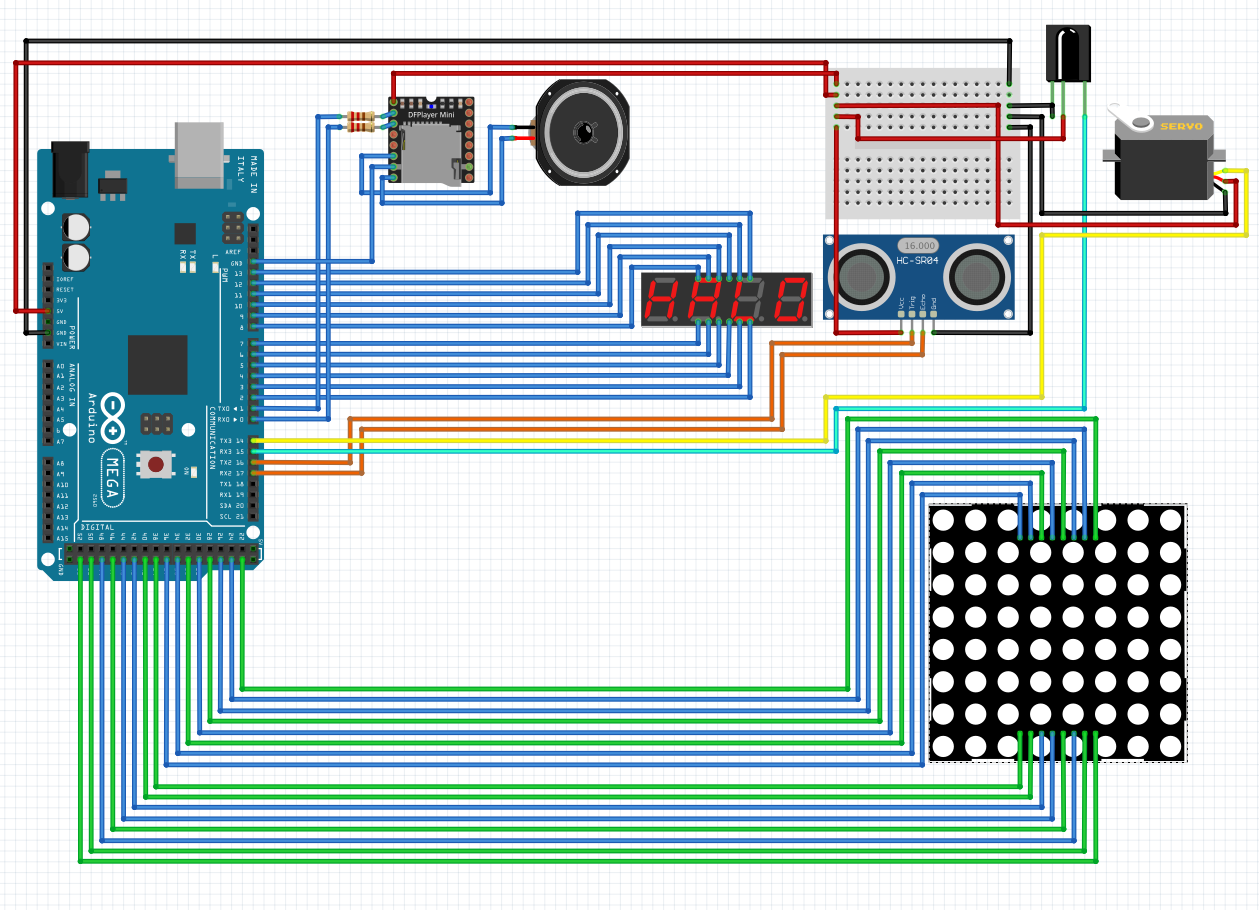
* De variabele *grootte* houdt bij hoeveel bits (blokken) in de huidige rij verplaatst moeten worden.
* Na elke knopdruk wordt de *grootte* herberekend op basis van de overblijvende blokken in de nieuwe gecombineerde rij.

### **Rij Verhoging:**

* De *row* variabele wordt verhoogd na elke succesvolle stapelactie, waardoor de blokken in de volgende rij worden geplaatst.
* Het spel gaat door totdat alle rijen zijn gebruikt of de blokken niet meer kunnen worden gestapeld.

## Blokschema





## Code

| #include <IRremote.h> #include <SoftwareSerial.h> #include "DFRobotDFPlayerMini.h" #include <Servo.h>  // Pins for 8x8 LED Matrix #define ROW\_1 A4 #define ROW\_2 3 #define ROW\_3 4 #define ROW\_4 5 #define ROW\_5 6 #define ROW\_6 7 #define ROW\_7 8 #define ROW\_8 9  #define COL\_1 10 #define COL\_2 11 #define COL\_3 12 #define COL\_4 13 #define COL\_5 A0 #define COL\_6 A1 #define COL\_7 A2 #define COL\_8 A3  int S1 = 2; int s1State = 0; bool flag = false; int grootte = 5; int row = 0; byte display\_row[8] = { B11111111, B11111111, B11111111, B11111111, B11111111, B11111111, B11111111, B11111111 };  float timeCount = 0; long time = millis(); int delayTime = 10;  unsigned long lastDebounceTime = 0; // the last time the output pin was toggled unsigned long debounceDelay = 50;   // Pins for Ultrasone Sensor const int trigPin = 22; const int echoPin = 23; const int ledPin = 52; // LED indicator pin  // Pins for DFPlayer Mini static const uint8\_t PIN\_MP3\_TX = 18; static const uint8\_t PIN\_MP3\_RX = 19; SoftwareSerial softwareSerial(PIN\_MP3\_RX, PIN\_MP3\_TX);  //object voor de player DFRobotDFPlayerMini player;  // IR Remote button codes const unsigned long POWER\_BUTTON\_CODE = 0xBD42FF00; const unsigned long VOLUME\_UP\_BUTTON\_CODE = 0xB946FF00; const unsigned long VOLUME\_DOWN\_BUTTON\_CODE = 0xEA15FF00; const unsigned long RESET\_BUTTON\_CODE = 0x49B0F624;  bool systemOn = true; // System state int volume = 15; // Volume level, default 15 (schaal van 0-30)  // Define IR receiver pin const int IR\_RECEIVER\_PIN = 50; IRrecv irReceiver(IR\_RECEIVER\_PIN);  // Servo Servo myservo; int pos = 0;  void setup() {  Serial.begin(9600);   // Setup 8x8 LED Matrix  const byte rows[] = { ROW\_1, ROW\_2, ROW\_3, ROW\_4, ROW\_5, ROW\_6, ROW\_7, ROW\_8 };  const byte col[] = { COL\_1, COL\_2, COL\_3, COL\_4, COL\_5, COL\_6, COL\_7, COL\_8 };   for (byte i = 0; i < 8; i++) {  pinMode(rows[i], OUTPUT);  pinMode(col[i], OUTPUT);  }  pinMode(S1, INPUT\_PULLUP);  attachInterrupt(digitalPinToInterrupt(S1), buttonPressed, FALLING);   // Setup Ultrasone Sensor  pinMode(trigPin, OUTPUT);  pinMode(echoPin, INPUT);  pinMode(ledPin, OUTPUT);   // Setup DFPlayer Mini  softwareSerial.begin(9600);  if (player.begin(softwareSerial)) {  Serial.println("DFPlayer Mini initialized.");  player.volume(volume); // Set default volume  } else {  Serial.println("Failed to initialize DFPlayer Mini.");  }   // Setup IR Receiver  irReceiver.enableIRIn();   // Setup Servo  myservo.attach(24);  myservo.write(pos); }  void loop() {  // Check for IR Remote commands  if (irReceiver.decode()) {  unsigned long receivedCode = irReceiver.decodedIRData.decodedRawData;   if (receivedCode == POWER\_BUTTON\_CODE) {  systemOn = !systemOn;  Serial.println(systemOn ? "System ON" : "System OFF");   if (!systemOn) {  powerDownSystem();  } else {  powerUpSystem();  }  } else if (receivedCode == VOLUME\_UP\_BUTTON\_CODE) {  if (systemOn && volume < 30) { // Max volume is 30  volume += 5; // Verhoog volume met sprongen van 5  updateVolume();  }  } else if (receivedCode == VOLUME\_DOWN\_BUTTON\_CODE) {  if (systemOn && volume > 0) {  volume -= 5; // Verlaag volume met sprongen van 5  updateVolume();  }  } else if (receivedCode == RESET\_BUTTON\_CODE) {  if (systemOn) {  resetGame(); // Reset de game als systeem aan staat  }  }   irReceiver.resume();  }   // TOWERGAME logic  if (!systemOn) {  return; // If system is off, do not run game logic  }   updateGameDisplay();  checkUltrasoundSensor(); }  void updateGameDisplay() {  // Game logic  bool flag = false;   for (int i = 0; i < grootte + 8; i++) {  if (!flag) {  if (i < grootte) {  display\_row[row] = display\_row[row] << 1;  } else {  byte x = B11111111;  display\_row[row] = ((display\_row[row] << 1) | (x >> (7 - (i - grootte))));  }  time = millis();  while (millis() - time < delayTime) {  drawScreen();  }  }  }   for (int i = grootte + 8; i > 0; i--) {  if (!flag) {  if (i > 8) {  display\_row[row] = display\_row[row] >> 1;  } else {  byte x = B11111111;  display\_row[row] = ((display\_row[row] >> 1) | (x << (i - 1)));  }  time = millis();  while (millis() - time < delayTime) {  drawScreen();  }  }  } }  void drawScreen() {  // Draw function for 8x8 LED Matrix  const byte rows[] = { ROW\_1, ROW\_2, ROW\_3, ROW\_4, ROW\_5, ROW\_6, ROW\_7, ROW\_8 };  const byte col[] = { COL\_1, COL\_2, COL\_3, COL\_4, COL\_5, COL\_6, COL\_7, COL\_8 };   for (byte o = 0; o < 8; o++) {  digitalWrite(rows[o], HIGH);  for (byte a = 0; a < 8; a++) {  digitalWrite(col[a], (display\_row[o] >> a) & 0x01);  delayMicroseconds(500);  digitalWrite(col[a], 1);  }  digitalWrite(rows[o], LOW);  } }  void buttonPressed() {  if (millis() - lastDebounceTime > debounceDelay){  lastDebounceTime = millis();  flag = true;   int previousRow = row; // Houd de vorige waarde van row bij   row++;   if (row > 1) {  display\_row[row - 1] = (display\_row[row - 2] | display\_row[row - 1]);  grootte = 0;  for (int i = 0; i < 8; i++) {  if (((display\_row[row - 1] >> i) & 0x01) == 0) {  grootte++;  }  }   // Controleer of er een nieuw blok is geplaatst  if (row > previousRow) {  player.playMp3Folder(1); // Play MP3 file 0001.mp3 in folder "mp3"  }   // Controleer of de toren de top heeft bereikt  if (row == 7) {  openServo();  }  }  } }  void checkUltrasoundSensor() {  digitalWrite(trigPin, LOW);  delayMicroseconds(2);  digitalWrite(trigPin, HIGH);  delayMicroseconds(10);  digitalWrite(trigPin, LOW);    long duration = pulseIn(echoPin, HIGH);  int distance = duration \* 0.034 / 2;    if (distance > 0 && distance < 2000) { // Controleer op geldige afstandsmeting  if (distance > 100 && distance < 800) {  Serial.println("LED AAN");  digitalWrite(ledPin, HIGH);  } else {  Serial.println("LED UIT");  digitalWrite(ledPin, LOW);  }  } else {  Serial.println("Foutieve meting, wordt genegeerd");  } }   void updateVolume() {  // Update volume function for DFPlayer Mini  player.volume(volume); }  void powerDownSystem() {  // Power down all pins except IR receiver pin  for (int pin = 2; pin <= 53; pin++) {  if (pin != IR\_RECEIVER\_PIN) {  pinMode(pin, INPUT);  }  }  for (int pin = A0; pin <= A15; pin++) {  pinMode(pin, INPUT);  }  Serial.println("System is powered down."); }  void powerUpSystem() {  // Power up all pins  for (int pin = 2; pin <= 53; pin++) {  pinMode(pin, OUTPUT);  }  for (int pin = A0; pin <= A15; pin++) {  pinMode(pin, OUTPUT);  }  setup(); // Reset game state  Serial.println("System is powered up."); }  void openServo() {  for (pos = 0; pos <= 90; pos++) {  myservo.write(pos);  delay(5);  }  for (pos = 90; pos >= 0; pos--) {  myservo.write(pos);  delay(5);  } }  void resetGame() {  grootte = 5; // Reset grootte naar oorspronkelijke waarde  row = 0; // Reset row naar oorspronkelijke waarde  for (int i = 0; i < 8; i++) {  display\_row[i] = B11111111; // Reset elke byte in display\_row naar B11111111  } } |
| --- |

### **Bronnen**

<https://projecthub.arduino.cc/SAnwandter1/programming-8x8-led-matrix-a3b852>

<https://github.com/Arduino-IRremote/Arduino-IRremote/tree/master/examples>

<https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/#google_vignette>