Hieronder is de code weergeven voor het gebruiken van de Ledmatrix.

Alle tekst waar een // voor staat is opmerking.

Tijdens de code is dit meerdere keren op bepaalde plekken aangegeven om duidelijkheid te geven, over een stukje code.

#include <EEPROM.h> // de EEProm bibliotheek aanroepen.

// Kolommen (negatief kathodes.)

int latchPin1 = 2; //Arduino pin naar pin 12, de RCK van register 74HC595.

int clockPin1 = 3; //Arduino pin naar pin 11, de clock van register 74HC595.

int dataPin1 = 4; //Arduino pin naar pin 14, de Data van register 74HC595.

// Rijen (positief anodes)

int latchPin2 = 5; // Arduino pin naar pin 12, de RCK van register 74HC595.

int clockPin2 = 6; // Arduino pin naar pin 11, de clock van register 74HC595.

int dataPin2 = 7; // Arduino pin naar pin 14, de Data van register 74HC595.

//Aanmaken van de bitmappen.

byte bitmap[8][5]; //het eerste getal is voor het aantal rijen en 2e getal is hoeveel matrixen.

// Hier worden 8 kolommen als 1 matrix aangemaakt, ook wel "zones" genoemd.

int numZones = sizeof(bitmap) / 8;

int maxZoneIndex = numZones-1;

int numCols = numZones \* 8;

// De letters klaar maken.

// hier onder worden alle letters en cijfers beschreven.

// de letters zijn op gemaakt in 5x7 matrixen zodat het goed leesbaar blijft.

// hij begint bij de spatie ( Zie ASCII tabel) en gaat daarna verder de hele tabel af.

// alle tekens tussen ASCII 32 en ASCII 125 zijn hier onder gemaakt.

byte alphabets[][5] = {

{0,0,0,0,0}, // spatie ASCII 32

{0,0,125,0,0}, // uitroepteken ASCII 33

{0,96,0,96,0}, // " ASCII 34

{20,127,20,127,20}, // # ASCII 35

{36,42,127,42,18}, // $ ASCII 36

{17,2,4,8,17}, // % ASCII 37

{54,73,85,34,5}, // & ASCII 38

{0,0,104,112,0}, // ' ASCII 39

{28,34,65}, // ( ASCII 40

{65,34,28}, // ) ASCII 41

{20,8,62,8,20}, // \* ASCII 42

{8,8,62,8,8}, // + ASCII 43

{0,0,5,6,0}, // , ASCII 44

{8,8,8,8,8}, // - ASCII 45

{0,0,1,0,0}, // . ASCII 46

{1,2,4,8,16}, // / ASCII 47

{62,69,73,81,62}, // 0 ASCII 48

{0,33,127,1,0}, // 1 ASCII 49

{33,67,69,73,49}, // 2 ASCII 50

{66,65,81,105,70}, // 3 ASCII 51

{12,20,36,127,4}, // 4 ASCII 52

{113,81,81,81,78}, // 5 ASCII 53

{30,41,73,73,6}, // 6 ASCII 54

{64,64,79,80,96}, // 7 ASCII 55

{54,73,73,73,54}, // 8 ASCII 56

{48,73,73,74,60}, // 9 ASCII 57

{0,0,54,54,0}, // : ASCII 58

{0,0,53,54,0}, // ; ASCII 59

{0,8,20,34,65}, // < ASCII 60

{20,20,20,20,20}, // = ASCII 61

{0,65,34,20,8}, // > ASCII 62

{32,64,69,72,48}, // ? ASCII 63

{38,73,77,65,62}, // @ ASCII 64

{31,36,68,36,31}, // A ASCII 65

{127,73,73,73,54}, // B ASCII 66

{62,65,65,65,34}, // C ASCII 67

{127,65,65,34,28}, // D ASCII 68

{127,73,73,65,65}, // E ASCII 69

{127,72,72,72,64}, // F ASCII 70

{62,65,65,69,38}, // G ASCII 71

{127,8,8,8,127}, // H ASCII 72

{0,65,127,65,0}, // I ASCII 73

{2,1,1,1,126}, // J ASCII 74

{127,8,20,34,65}, // K ASCII 75

{127,1,1,1,1}, // L ASCII 76

{127,32,16,32,127}, // M ASCII 77

{127,32,16,8,127}, // N ASCII 78

{62,65,65,65,62}, // O ASCII 79

{127,72,72,72,48}, // P ASCII 80

{62,65,69,66,61}, // Q ASCII 81

{127,72,76,74,49}, // R ASCII 82

{50,73,73,73,38}, // S ASCII 83

{64,64,127,64,64}, // T ASCII 84

{126,1,1,1,126}, // U ASCII 85

{124,2,1,2,124}, // V ASCII 86

{126,1,6,1,126}, // W ASCII 87

{99,20,8,20,99}, // X ASCII 88

{96,16,15,16,96}, // Y ASCII 89

{67,69,73,81,97}, // Z ASCII 90

{0,127,65,65,0}, // [ ASCII 91

{0,0,0,0,0}, // \ ASCII 92

{0,65,65,127,0}, // ] ASCII 93

{16,32,64,32,16}, // ^ ASCII 94

{1,1,1,1,1}, // \_ ASCII 95

{0,64,32,16,0}, // ` ASCII 96

{2,21,21,21,15}, // a ASCII 97

{127,5,9,9,6}, // b ASCII 98

{14,17,17,17,2}, // c ASCII 99

{6,9,9,5,127}, // d ASCII 100

{14,21,21,21,12}, // e ASCII 101

{8,63,72,64,32}, // f ASCII 102

{24,37,37,37,62}, // g ASCII 103

{127,8,16,16,15}, // h ASCII 104

{0,0,47,0,0}, // i ASCII 105

{2,1,17,94,0}, // j ASCII 106

{127,4,10,17,0}, // k ASCII 107

{0,65,127,1,0}, // l ASCII 108

{31,16,12,16,31}, // m ASCII 109

{31,8,16,16,15}, // n ASCII 110

{14,17,17,17,14}, // o ASCII 111

{31,20,20,20,8}, // p ASCII 112

{8,20,20,12,31}, // q ASCII 113

{31,8,16,16,8}, // r ASCII 114

{2,21,21,21,9}, // s ASCII 115

{16,126,17,1,2}, // t ASCII 116

{30,1,1,2,31}, // u ASCII 117

{28,2,1,2,28}, // v ASCII 118

{30,1,6,1,30}, // w ASCII 119

{17,10,4,10,17}, // x ASCII 120

{24,5,5,5,30}, // y ASCII 121

{17,19,21,25,17}, // z ASCII 122

{0,0,8,54,65}, // { ASCII 123

{0,0,127,0,0}, // | ASCII 124

{65,54,8,0,0}, // } ASCII 125

};

// Hier wordt alles op gemaakt voor gebruik.

// Alle aansluitingen die boven aangegeven waren, worden nu instelt als uitgangen.

void setup() {

pinMode(latchPin1, OUTPUT);

pinMode(clockPin1, OUTPUT);

pinMode(dataPin1, OUTPUT);

pinMode(latchPin2, OUTPUT);

pinMode(clockPin2, OUTPUT);

pinMode(dataPin2, OUTPUT);

// Hier wordt de communicatie tussen computer en arduino board aangeroepen.

Serial.begin(9600);

// Hier onder worden de "zones" leeg gemaakt.

for (int row = 0; row < 8; row++) {

for (int zone = 0; zone <= maxZoneIndex; zone++) {

bitmap[row][zone] = 0;

}

}

}

// Hier onder worden de voorbereidingen getroffen om alle data te kunnen verzenden, en te kunnen weergeven.

void RefreshDisplay()

{

for (int row = 0; row < 8; row++) {

int rowbit = 1 << row;

rowbit = ~rowbit;

// Hier zetten we de latchpin laag zolang er iets gestuurd wordt.

digitalWrite(latchPin2, LOW);

// Hier word de Data verstuurd naar de shift registers.

shiftOut(dataPin2, clockPin2, MSBFIRST, rowbit);

// Hier houden we de latchpin laag zolang er iets gestuurd wordt.

digitalWrite(latchPin1, LOW);

for (int zone = maxZoneIndex; zone >= 0; zone--) {

shiftOut(dataPin1, clockPin1, MSBFIRST, bitmap[row][zone]);

}

// Pinnen worden te gelijk weer "hoog" gezet, waardoor je geen contact dender krijgt.

// Hierdoor weet ook het boardje, dat hij niet meer hoeft te luisteren naar data.

digitalWrite(latchPin1, HIGH);

digitalWrite(latchPin2, HIGH);

// Kleine wachttijd zodat wij als mensen het kunnen zien.

delayMicroseconds(500);

}

}

// Hier worden de rijen en kolommen om gezet naar 1 ledje, zodat je ze aan en uit kan zetten.

void Plot(int col, int row, bool isOn)

{

int zone = col / 8;

int colBitIndex = col % 8;

byte colBit = 1 << colBitIndex;

// Als je hieronder de **& (~colBit)** omdraait met **| colBit** dan zet je de achtergrond rood en letters //zwart.

if (isOn)

bitmap[row][zone] = bitmap[row][zone] | colBit;

else

bitmap[row][zone] = bitmap[row][zone] & (~colBit);

}

// Hieronder wordt de functie geschreven die we later gebruiken.

//de tekst van de computer wordt omgezet naar de "zones" en worden de letters omgezet naar een // 1 of 0 voor de ledjes.

void AlphabetSoup()

{

#define INLENGTH 20 // Hier maak je het totaal aantal tekens wat gebruikt mag worden.

#define INTERMINATOR 13 // Bepaling om te kijken of de invoer klaar is.

char msg[INLENGTH+1];

char tmpMsg[INLENGTH+1];

int inCount; // Dit maakt de geschiedenis leeg.

for (int charIndex=0; charIndex < (sizeof(tmpMsg)-1); charIndex++)

{

tmpMsg[charIndex] = 0;

}

inCount = 0;

do {

tmpMsg[inCount] = Serial.read(); // lees serial uit.

if (tmpMsg[inCount] == INTERMINATOR) break;

} while (++inCount < INLENGTH);

// Zodra er een teken hoger dan ASCII 30 wordt gebruikt, zal hij de geschiedenis overschrijven.

if (tmpMsg[0] > 30) {

// Bovenin hebben wij al aangegeven dat de ASCII tabel met tekens begint bij 32, dus je zal altijd

// overschrijven op het moment dat je iets invoert.

for (int charIndex=0; charIndex < (sizeof(tmpMsg)-1); charIndex++)

{

msg[charIndex] = tmpMsg[charIndex];

// vanaf hier schrijf je naar de EEPROM.

for (int i = 0; i < (sizeof(msg)-1); i++)

{

EEPROM.write(i, msg[i]);

}

}

}

// Hier haalt hij de gegevens op van de EEProm en schrijft ze weer weg in een variable.

char restored[20] ;

for (int i = 0; i < (sizeof(restored)-1); i++)

{

restored[i] = EEPROM.read(i);

}

for (int charIndex=0; charIndex < (sizeof(restored)-1); charIndex++)

{

// Tussen de ' ' meld je de code waar hij moet beginnen met tellen.

int alphabetIndex = restored[charIndex] - ' ';

// Bovenin hebben wij gezegd bij de letters dat we beginnen met de spatie.

if (alphabetIndex < 0) alphabetIndex=0;

// Hieronder worden de letters omgezet naar de juiste 5x7 aanduiding en begint de tekst op de // matrixen te scrollen.

// De letters zijn 5 ledjes breed maar wij nemen hier onder 6 omdat je dan wat ruimte krijgt tussen // de letters.

for (int col = 0; col < 6; col++)

{

for (int row = 0; row < 8; row++)

{

bool isOn = 0;

if (col<5) isOn = bitRead( alphabets[alphabetIndex][col], 7-row ) == 1;

// We tekenen de ledjes altijd de rechter matrix, en door de loop scrolt het naar links.

Plot( numCols-1, row, isOn); }

// Als je de 10 veranderd in een hoger getal, zal het langzamer gaan lopen. Als je sneller wil, // verander je het naar lager dan 10.

for (int refreshCount=0; refreshCount < 10; refreshCount++)

RefreshDisplay();

// Hier schuift de bitmap 1 kolom naar links.

for (int row=0; row<8; row++)

{

for (int zone=0; zone < numZones; zone++)

{

bitmap[row][zone] = bitmap[row][zone] >> 1;

// Hier overschrijft hij elke keer de rij links van de matrix.

if (zone < maxZoneIndex) bitWrite(bitmap[row][zone], 7, bitRead(bitmap[row][zone+1],0));

}

}

}

}

}

// Alles is nu ingesteld en laat je het programma alleen maar opnieuw starten.

// Hij laad alleen het laatste deel, zodat de rest niet overschreven wordt.

void loop() {

AlphabetSoup();

}