/\*  
 \* Eindopdracht - Show Your Moves  
 \*   
 \* Eigenaar:  
 \* Maarten Paauw <s1094220@student.hsleiden.nl>  
 \* s1094220  
 \* INF3C  
 \*   
 \* Versie: 1  
 \* Aangemaakt: 22 maart 2018  
 \* Gewijzigd: 31 maart 2018  
 \*/  
  
#include <avr/io.h>  
#include <stdlib.h>  
#include <stdio.h>  
#include <util/delay.h>  
#include <avr/interrupt.h>  
  
// USART wordt gebruikt voor het debuggen van de sensor data.  
#include <USART.h>  
  
// De pin waar de DHT11 data over verstuurd wordt.  
#define DHT11\_PIN PD6  
  
// De bank waar de DHT11 data over verstuurd wordt.  
#define DHT11\_BANK DDRD  
  
// De port waar de DHT11 data over verstuurd wordt.  
#define DHT11\_PORT PORTD  
  
// De bank waar de 8x8 matrix pinnen op aangesloten zitten.  
#define MATRIX\_BANK DDRC  
  
// De port waar de 8x8 matrix pinnen op aangesloten zitten.  
#define MATRIX\_PORT PORTC  
  
// De port en pin voor de SER pin op de shift register.  
#define SER\_PIN PC0  
  
// De port en pin voor de RCLK pin op de shift register.  
#define RCLK\_PIN PC1  
  
// De port en pin voor de SRCLK pin op de shift register.  
#define SRCLK\_PIN PC2  
  
// Count variabele  
int count = 0;  
  
// Variabele om tijdelijk de data (enkele bit) van de sensor op te slaan.  
uint8\_t temporarily = 0;  
  
// Variabele om de integraal van de luchtvochtigheid op te slaan.  
uint8\_t integral\_humidity;  
  
// Variabele om de decimaal van de luchtvochtigheid op te slaan.  
uint8\_t decimal\_humidity;  
  
// Variabele om de integraal van de temperatuur (in celcius) op te slaan.  
uint8\_t integral\_temperature;  
  
// Variabele om de decimaal van de temperatuur (in celcius) op te slaan.  
uint8\_t decimal\_temperature;  
  
// Variabele om de checkcum over de 4 eerder genoemde data op te slaan.  
uint8\_t checksum;  
  
// Initialiseer de DHT11 sensor.  
void init\_DHT11()  
{  
 // In de datasheet is er uitgelegd dat de DHT11 sensor 1 seconde nodig  
 // heeft met opstarten. Daarom even een delay van 1 seconde.  
 \_delay\_ms(1000);  
}  
  
// De DHT11 sensor moet 1 seconden wachten voor het meten van  
// nieuwe data. Deze tijd komt uit de datasheet.  
int DHT11\_measure\_time()  
{  
 // (1000 / (16000000 / 1024)) \* 1000 = 64;  
 return 64;  
}  
  
// Wacht net zolang dat de DHT11 pin clear is.  
void loop\_until\_dht\_is\_clear()  
{  
 // Wacht net zolang dat de DHT11 pin clear is.  
 loop\_until\_bit\_is\_clear(PIND, DHT11\_PIN);  
}  
  
// Wacht net zolang dat de DHT11 pin geset is.  
void loop\_until\_dht\_is\_set()  
{  
 // Wacht net zolang dat de DHT11 pin geset is.  
 loop\_until\_bit\_is\_set(PIND, DHT11\_PIN);  
}  
  
// Functie om de request uit te voeren richting de DHT11 sensor.  
void request()  
{  
 // Geef aan dat de DHT11 pin op de D bank gebruikt gaat worden.  
 DHT11\_BANK |= (1 << DHT11\_PIN);  
  
 // Zet de DHT11 pin op 0. Dit heb ik in mijn vorige verslag beschreven als  
 // een "pull down".  
 DHT11\_PORT &= ~(1 << DHT11\_PIN);  
  
 // De "pull down" moet minimaal 18 miliseconden plaats vinden.  
 // Ik hou het bij de minimale "pull down" tijd.  
 \_delay\_ms(18);  
  
 // Zet de DHT11 pin op 1. Dit heb ik in mijn vorige verslag beschreven als  
 // een "pull up".  
 DHT11\_PORT |= (1 << DHT11\_PIN);  
  
 // De "pull up" moet exact 40 microseconden plaats vinden.  
 \_delay\_us(40);  
}  
  
// Functie die de (automatische) response van de DHT11 sensor afhandelt.  
void response()  
{  
 // Zet de DHT11 pin op 0. Nu kan de DHT11 sensor zijn response geven.  
 DHT11\_BANK &= ~(1 << DHT11\_PIN);  
  
 // Loop net zolang dat de DHT11 pin clear is.  
 loop\_until\_dht\_is\_clear();  
  
 // Loop net zolang dat de DHT11 pin geset is.  
 // Dit is de 54 microseconden die ik in de uitleg aangeef.  
 loop\_until\_bit\_is\_set(PIND, DHT11\_PIN);  
  
 // Loop net zolang dat de DHT11 pin clear is.  
 // Dit is de 80 microseconden die ik in de uitleg aangeef.  
 loop\_until\_dht\_is\_clear();  
}  
  
// Verkrijg de 8-bit data van de sensor.  
uint8\_t receive\_data()  
{  
 // Elk stuk data bestaat uit 8-bit. Daarom wordt er 8x een bit uitgelezen.  
 for (int q = 0; q < 8; q++)  
 {  
 // Loop net zolang dat de DHT11 pin clear is.  
 // Dit is de 54 microseconden de ik in de uitleg aangeef.  
 loop\_until\_dht\_is\_clear();  
  
 // Loop net zolang dat de DHT11 pin geset is.  
 loop\_until\_bit\_is\_set(PIND, DHT11\_PIN);  
  
 // Als de bit 0 moet zijn wordt er voor 24 microseconden gewacht.  
 // Als de bit 1 moet zijn wordt er voor 70 microseconden gewacht.  
 // Ik wil precies op de helft van de tijd tussen low en high wachten.  
 // Op dat moment kan je uitlezen of de pin high of low staat.  
 \_delay\_us(24 + (70 / 2));  
  
 // Kijk of de DHT11 pin op high staat wordt er een 1 bedoelt als bit.  
 if (PIND & (1 << DHT11\_PIN))  
 {  
 // Schuif de vorige waarde 1 plek naar links en sla een nieuwe 1 op  
 // in het binaire getal.  
 temporarily = (temporarily << 1) | (0x01);  
 }  
  
 // Als de pin op low staat wordt er een 0 bedoelt als bit.  
 else  
 {  
 // Schuif de vorige waarde 1 plek naar links. Automatisch wordt er  
 // aan de rechter kant een nul toegevoegd.  
 temporarily = (temporarily << 1);  
 }  
  
 // Loop net zolang dat de DHT11 op hoog staat.  
 // Dit is de 54 microseconden de ik in de uitleg aangeef.  
 loop\_until\_dht\_is\_clear();  
 }  
  
 // Geef het binaire getal terug.  
 return temporarily;  
}  
  
// Sla alle data uit de sensor uit.  
void save\_data()  
{  
 // Sla de integraal van de luchtvochtigheid op.  
 integral\_humidity = receive\_data();  
  
 // Sla de decimaal van de luchtvochtigheid op.  
 decimal\_humidity = receive\_data();  
  
 // Sla de integraal van de temperatuur op.  
 integral\_temperature = receive\_data();  
  
 // Sla de decimaal van de temperatuur op.  
 decimal\_temperature = receive\_data();  
  
 // Sla de checksum op.  
 checksum = receive\_data();  
}  
  
// Valideer de sensor data door middel van de checksum.  
uint8\_t validate\_sensor\_data()  
{  
 // Tel de luchtvochtigheid en de temeratuur bij elkaar op.  
 uint8\_t combined = integral\_humidity +  
 decimal\_humidity +  
 integral\_temperature +  
 decimal\_temperature;  
  
 // Geef terug dat de data klopt.  
 return combined == checksum;  
}  
  
// Print de sensor data uit.  
void debug(int sensor\_data, char extra\_string[])  
{  
 // placeholder voor de data voor het printen.  
 char data[5];  
  
 // Formateer de integer naar een string.  
 itoa(sensor\_data, data, 10);  
  
 // Toon de sensor data.  
 printString(data);  
  
 // Voeg de extra string toe.  
 printString(extra\_string);  
}  
  
// Initialiseer de 8x8 matrix.  
void init\_8x8\_matrix()  
{  
 // Defineer de pinnen die nodig zijn.  
 MATRIX\_BANK = (1 << SER\_PIN) | (1 << RCLK\_PIN) | (1 << SRCLK\_PIN);  
  
 // Zet alle C pinnen op low.  
 MATRIX\_PORT = 0b00000000;  
}  
  
// Zet te serial pin laag.  
void ser\_pin\_low()  
{  
 // Zet te serial pin laag.  
 MATRIX\_PORT &= ~(1 << SER\_PIN);  
}  
  
// Zet te serial pin hoog.  
void ser\_pin\_high()  
{  
 // Zet te serial pin hoog.  
 MATRIX\_PORT |= (1 << SER\_PIN);  
}  
  
// Zet te rclk pin laag.  
void rclk\_pin\_low()  
{  
 // Zet te rclk pin laag.  
 MATRIX\_PORT &= ~(1 << RCLK\_PIN);  
}  
  
// Zet te rclk pin hoog.  
void rclk\_pin\_high()  
{  
 // Zet te rclk pin hoog.  
 MATRIX\_PORT |= (1 << RCLK\_PIN);  
}  
  
// Zet te srclk pin laag.  
void srclk\_pin\_low()  
{  
 // Zet te srclk pin laag.  
 MATRIX\_PORT &= ~(1 << SRCLK\_PIN);  
}  
  
// Zet te srclk pin hoog.  
void srclk\_pin\_high()  
{  
 // Zet te srclk pin hoog.  
 MATRIX\_PORT |= (1 << SRCLK\_PIN);  
}  
  
// Zet iets op het display.  
void turn\_on\_dot (int row, int column)  
{  
 // Zet de srclk pin op laag.  
 srclk\_pin\_low();  
  
 // Zet de rclk pin op laag.  
 rclk\_pin\_low();  
  
 // Trek 1 van de rij af.  
 int real\_row = row - 1;  
  
 // Trek 1 van de kolom af.  
 int real\_column = column - 1;  
  
 // Loop door 8 bits heen  
 for (int i = 0; i < 8; i++)   
 {  
 // Controleer of de bit hoog moet zijn  
 if (i == real\_row)  
 {  
 // Zet de SER pin op hoog.  
 ser\_pin\_high();  
 }  
  
 // Zo niet;  
 else  
 {  
 // Zet de SER pin op laag.  
 ser\_pin\_low();  
 }  
  
 // Zet de RCLK pin op hoog.  
 srclk\_pin\_high();  
  
 // Zet de RCLK pin op laag.  
 srclk\_pin\_low();  
 }  
  
 // Loop door 8 bits heen  
 for (int i = 0; i < 8; i++)   
 {  
 // Controleer of de bit hoog moet zijn  
 if (i == real\_column)  
 {  
 // Zet de SER pin op hoog.  
 ser\_pin\_low();  
 }  
  
 // Zo niet;  
 else  
 {  
 // Zet de SER pin op laag.  
 ser\_pin\_high();  
 }  
  
 // Zet de RCLK pin op hoog.  
 srclk\_pin\_high();  
  
 // Zet de RCLK pin op laag.  
 srclk\_pin\_low();  
 }  
  
 // Zet de srclk pin op hoog.  
 rclk\_pin\_high();  
}  
  
// Show sad smiley.  
void matrix\_smiley\_sad()  
{  
 // Rijen en kolommen die aan moeten voor het tonen van een zielig gezicht.  
 uint8\_t dots [30][2] = {  
 {1, 2},  
 {1, 3},  
 {1, 4},  
 {1, 5},  
 {1, 6},  
 {1, 7},  
 {2, 1},  
 {2, 8},  
 {3, 1},  
 {3, 3},  
 {3, 6},  
 {3, 8},  
 {4, 1},  
 {4, 8},  
 {5, 1},  
 {5, 4},  
 {5, 5},  
 {5, 8},  
 {6, 1},  
 {6, 3},  
 {6, 6},  
 {6, 8},  
 {7, 1},  
 {7, 8},  
 {8, 2},  
 {8, 3},  
 {8, 4},  
 {8, 5},  
 {8, 6},  
 {8, 7},  
 };  
  
 // Loop 30 keer (zoveel leds zijn er nodig).  
 for (int i = 0; i < 30; i++)  
 {  
 // Zet de specifieke led aan.  
 turn\_on\_dot(dots[i][0], dots[i][1]);  
  
 // Laat de led heel even aan staan.  
 \_delay\_us(100);  
 }  
}  
  
// Show happy smiley.  
void matrix\_smiley\_happy()  
{  
 // Rijen en kolommen die aan moeten voor het tonen van een blij gezicht.  
 uint8\_t dots [30][2] = {  
 {1, 2},  
 {1, 3},  
 {1, 4},  
 {1, 5},  
 {1, 6},  
 {1, 7},  
 {2, 1},  
 {2, 8},  
 {3, 1},  
 {3, 3},  
 {3, 6},  
 {3, 8},  
 {4, 1},  
 {4, 8},  
 {5, 1},  
 {5, 3},  
 {5, 6},  
 {5, 8},  
 {6, 1},  
 {6, 4},  
 {6, 5},  
 {6, 8},  
 {7, 1},  
 {7, 8},  
 {8, 2},  
 {8, 3},  
 {8, 4},  
 {8, 5},  
 {8, 6},  
 {8, 7},  
 };  
  
 // Loop 30 keer (zoveel leds zijn er nodig).  
 for (int i = 0; i < 30; i++)  
 {  
 // Zet de specifieke led aan.  
 turn\_on\_dot(dots[i][0], dots[i][1]);  
  
 // Laat de led heel even aan staan.  
 \_delay\_us(100);  
 }  
}  
  
// Overflow timer.  
void init\_timer\_overflow()  
{  
 // Timer mask 0.  
 TIMSK0 |= (1 << TOIE0);  
  
 // Timer mask 1.  
 TIMSK1 |= (1 << TOIE1);  
  
 // Timer 0 instellingen.  
 TCCR0B |= (1 << CS00) | (1 << CS02);  
  
 // Timer 1 instellingen.  
 TCCR1B |= (1 << CS10);  
  
 // Zet de interrupts aan.  
 sei();  
}  
  
// Functie voor het controleren of er tussen 2 waardes in zit.  
uint8\_t between(int value, int min, int max)  
{  
 // Kijk of het tussen 2 waardes in ligt.  
 return (value <= max && value >= min);  
}  
  
// Functie om te controleren of de temeratuur en de luchtvochtigheid perfect zijn.  
uint8\_t right\_conditions ()  
{  
 // Geef het terug.  
 return (between(integral\_temperature, 18, 22) && between(integral\_humidity, 40, 60));  
}  
  
// Timer overflow interrupt.  
ISR (TIMER0\_OVF\_vect)  
{  
 // Willen maar 1 keer meten per 10 seconden.  
 int count\_check = DHT11\_measure\_time() \* 10;  
  
 // Is het tijd voor een nieuwe meting?  
 if (count > count\_check)  
 {  
 // Zet de count terug op 0.  
 count = 0;  
  
 // Geef aan de DHT11 sensor door dat je een request wilt gaan doen.  
 request();  
  
 // Handel de (automatische) response van de DHT11 sensor af.  
 response();  
  
 // Sla de data van de sensor op in variabelen.  
 save\_data();  
  
 // Controleer of de DHT sensor data valide is.  
 if (validate\_sensor\_data())  
 {  
 // Print een titel.  
 printString("Luchtvochtigheid: ");  
  
 // Print de luchtvochtigheid.  
 debug(integral\_humidity, ".");  
 debug(decimal\_humidity, "%");  
  
 // Print een enter.  
 printString("  
");  
  
 // Print een titel.  
 printString("Temperatuur: ");  
  
 // Print de temperatuur.  
 debug(integral\_temperature, ".");  
 debug(decimal\_temperature, "°C");  
  
 // Print twee enters.  
 printString("  
  
");  
 }  
 }  
  
 // Zo niet;  
 else  
 {  
 // Tel de count 1 op.  
 count++;  
 }  
}  
  
// Functie on de matrix te bedienen.  
ISR (TIMER1\_OVF\_vect)  
{  
 // Bereken de temperatuur.  
 if (right\_conditions())  
 {  
 // Blije smiley.  
 matrix\_smiley\_happy();  
 }  
  
 // Niet de juiste omstandigheden.  
 else  
 {  
 // Verdrietige smiley.  
 matrix\_smiley\_sad();  
 }  
}  
  
// De main functie.  
int main(void)  
{  
 // Initialiseer de seriele verbinding voor het schrijven.  
 initUSART();  
  
 // Initialiseer de DHT11 sensor.  
 init\_DHT11();  
  
 // Initialiseer de 8x8 matrix.  
 init\_8x8\_matrix();  
  
 // Initialiseer de overflow timer.  
 init\_timer\_overflow();  
  
 // Loop voor altijd.  
 while (1) { }  
}