KBS Microcontrollers

Verslag Opdracht 1

Door:

Daan Stout

S1076173

Donovan Walter

S1078826

Rick Winters

S1080129

De code die we hebben gebruikt:

LCDTest.ino:

|  |
| --- |
| #include <avr/io.h>  #include <avr/interrupt.h>  #include <stdint.h>  #include <Wire.h>  #include <SPI.h>  #include <digitalWriteFast.h>  #include <GraphicsLib.h>  #include <MI0283QT9.h>  #include <Arduino.h>  #include "nunchuck\_funcs.h"  MI0283QT9 lcd;  volatile uint16\_t count = 0;  volatile uint16\_t teller = 0;  ISR(TIMER2\_OVF\_vect) {  teller++;  if(teller >= 512){  count++;    teller = 0;  }  }  void tekenAssen(){  lcd.drawLine(lcd.getWidth()/2, 0, lcd.getWidth()/2, lcd.getHeight(), RGB(255, 0, 0)); //verticale lijn  lcd.drawLine(0, lcd.getHeight()/2, lcd.getWidth(), lcd.getHeight()/2, RGB(255,0,0)); // horizontale lijn  }  int main(){  init();    TCCR2A |= (1 << CS02) | (1 << CS00);  TIMSK2 |= (1 << TOIE0);      TCNT2 = 0;  sei();  byte accx,accy,zbut,cbut,x,y, x1, y1;  uint8\_t tp\_last\_x;  uint8\_t tp\_last\_y;  char c[128];    lcd.begin();  nunchuck\_setpowerpins();  nunchuck\_init();    //lcd.touchRead();  //lcd.touchStartCal(); //calibreren  lcd.fillScreen(RGB(255,255,255)); // scherm leeg  tekenAssen();      while(1){  nunchuck\_get\_data();    accx = nunchuck\_accelx(); // ranges from approx 70 - 182  accy = nunchuck\_accely(); // ranges from approx 65 - 173  zbut = nunchuck\_zbutton();  cbut = nunchuck\_cbutton();  x = nunchuck\_joyx();  y = nunchuck\_joyy();    if(x > 30 && x < 50){  x1 = lcd.getWidth()/4;  y1= lcd.getHeight()/2;  }  if(x > 200 && x < 255){  x1 = lcd.getWidth()/4 + lcd.getWidth()/2;  y1 = lcd.getHeight()/2;  }  if(y > 30 && y < 50){  y1 = lcd.getHeight()/4 + lcd.getHeight()/2;  x1 = lcd.getWidth()/2;  }  if(y > 200 && y < 255){  y1 = lcd.getHeight()/4;  x1 = lcd.getWidth()/2;  }  if((x > 120 && x < 140) && (y > 120 && y < 140)){  y1 = lcd.getHeight()/2;  x1 = lcd.getWidth()/2;  }      lcd.drawInteger(5, 5, count, DEC, RGB(0,0,0), RGB(255,255,255), 1);  if(tp\_last\_x != x1 || tp\_last\_y != y1){  lcd.fillCircle(tp\_last\_x, tp\_last\_y, 3, RGB(255,255,255));  }    sprintf(c, "X:%03i Y:%03i Z:%03i", x, y, lcd.touchZ()); //coordinaten opmaak  lcd.drawText(100, 2, c, RGB(0,0,0), RGB(255,255,255), 1); //coordinaten op lcd tekenen  tekenAssen();  lcd.fillCircle(x1, y1, 2, RGB(0,0,255)); //cirkel tekenen  tp\_last\_x = x1;  tp\_last\_y = y1;  }  return 0;  } |

Nunchuck\_funcs.h:

|  |
| --- |
| /\*  \* Nunchuck functions -- Talk to a Wii Nunchuck  \*  \* This library is from the Bionic Arduino course :  \* http://todbot.com/blog/bionicarduino/  \*  \* 2007-11 Tod E. Kurt, http://todbot.com/blog/  \*  \* The Wii Nunchuck reading code originally from Windmeadow Labs  \* http://www.windmeadow.com/node/42  \*/  #if (ARDUINO >= 100)  #include <Arduino.h>  #else  #include <WProgram.h>  //#define Wire.write(x) Wire.send(x)  //#define Wire.read() Wire.receive()  #endif  static uint8\_t nunchuck\_buf[6]; // array to store nunchuck data,  // Uses port C (analog in) pins as power & ground for Nunchuck  static void nunchuck\_setpowerpins()  {  #define pwrpin PORTC3  #define gndpin PORTC2  DDRC |= \_BV(pwrpin) | \_BV(gndpin);  PORTC &=~ \_BV(gndpin);  PORTC |= \_BV(pwrpin);  delay(100); // wait for things to stabilize  }  // initialize the I2C system, join the I2C bus,  // and tell the nunchuck we're talking to it  static void nunchuck\_init()  {  Wire.begin(); // join i2c bus as master  Wire.beginTransmission(0x52);// transmit to device 0x52  #if (ARDUINO >= 100)  Wire.write((uint8\_t)0x40);// sends memory address  Wire.write((uint8\_t)0x00);// sends sent a zero.  #else  Wire.send((uint8\_t)0x40);// sends memory address  Wire.send((uint8\_t)0x00);// sends sent a zero.  #endif  Wire.endTransmission();// stop transmitting  }  // Send a request for data to the nunchuck  // was "send\_zero()"  static void nunchuck\_send\_request()  {  Wire.beginTransmission(0x52);// transmit to device 0x52  #if (ARDUINO >= 100)  Wire.write((uint8\_t)0x00);// sends one byte  #else  Wire.send((uint8\_t)0x00);// sends one byte  #endif  Wire.endTransmission();// stop transmitting  }  // Encode data to format that most wiimote drivers except  // only needed if you use one of the regular wiimote drivers  static char nunchuk\_decode\_byte (char x)  {  x = (x ^ 0x17) + 0x17;  return x;  }  // Receive data back from the nunchuck,  // returns 1 on successful read. returns 0 on failure  static int nunchuck\_get\_data()  {  int cnt=0;  Wire.requestFrom (0x52, 6);// request data from nunchuck  while (Wire.available ()) {  // receive byte as an integer  #if (ARDUINO >= 100)  nunchuck\_buf[cnt] = nunchuk\_decode\_byte( Wire.read() );  #else  nunchuck\_buf[cnt] = nunchuk\_decode\_byte( Wire.receive() );  #endif  cnt++;  }  nunchuck\_send\_request(); // send request for next data payload  // If we recieved the 6 bytes, then go print them  if (cnt >= 5) {  return 1; // success  }  return 0; //failure  }  // Print the input data we have recieved  // accel data is 10 bits long  // so we read 8 bits, then we have to add  // on the last 2 bits. That is why I  // multiply them by 2 \* 2  static void nunchuck\_print\_data()  {  static int i=0;  int joy\_x\_axis = nunchuck\_buf[0];  int joy\_y\_axis = nunchuck\_buf[1];  int accel\_x\_axis = nunchuck\_buf[2]; // \* 2 \* 2;  int accel\_y\_axis = nunchuck\_buf[3]; // \* 2 \* 2;  int accel\_z\_axis = nunchuck\_buf[4]; // \* 2 \* 2;  int z\_button = 0;  int c\_button = 0;  // byte nunchuck\_buf[5] contains bits for z and c buttons  // it also contains the least significant bits for the accelerometer data  // so we have to check each bit of byte outbuf[5]  if ((nunchuck\_buf[5] >> 0) & 1)  z\_button = 1;  if ((nunchuck\_buf[5] >> 1) & 1)  c\_button = 1;  if ((nunchuck\_buf[5] >> 2) & 1)  accel\_x\_axis += 1;  if ((nunchuck\_buf[5] >> 3) & 1)  accel\_x\_axis += 2;  if ((nunchuck\_buf[5] >> 4) & 1)  accel\_y\_axis += 1;  if ((nunchuck\_buf[5] >> 5) & 1)  accel\_y\_axis += 2;  if ((nunchuck\_buf[5] >> 6) & 1)  accel\_z\_axis += 1;  if ((nunchuck\_buf[5] >> 7) & 1)  accel\_z\_axis += 2;  Serial.print(i,DEC);  Serial.print("\t");  Serial.print("joy:");  Serial.print(joy\_x\_axis,DEC);  Serial.print(",");  Serial.print(joy\_y\_axis, DEC);  Serial.print(" \t");  Serial.print("acc:");  Serial.print(accel\_x\_axis, DEC);  Serial.print(",");  Serial.print(accel\_y\_axis, DEC);  Serial.print(",");  Serial.print(accel\_z\_axis, DEC);  Serial.print("\t");  Serial.print("but:");  Serial.print(z\_button, DEC);  Serial.print(",");  Serial.print(c\_button, DEC);  Serial.print("\r\n"); // newline  i++;  }  // returns zbutton state: 1=pressed, 0=notpressed  static int nunchuck\_zbutton()  {  return ((nunchuck\_buf[5] >> 0) & 1) ? 0 : 1; // voodoo  }  // returns zbutton state: 1=pressed, 0=notpressed  static int nunchuck\_cbutton()  {  return ((nunchuck\_buf[5] >> 1) & 1) ? 0 : 1; // voodoo  }  // returns value of x-axis joystick  static int nunchuck\_joyx()  {  return nunchuck\_buf[0];  }  // returns value of y-axis joystick  static int nunchuck\_joyy()  {  return nunchuck\_buf[1];  }  // returns value of x-axis accelerometer  static int nunchuck\_accelx()  {  return nunchuck\_buf[2]; // FIXME: this leaves out 2-bits of the data  }  // returns value of y-axis accelerometer  static int nunchuck\_accely()  {  return nunchuck\_buf[3]; // FIXME: this leaves out 2-bits of the data  }  // returns value of z-axis accelerometer  static int nunchuck\_accelz()  {  return nunchuck\_buf[4]; // FIXME: this leaves out 2-bits of the data  } |

Wij hebben niet gebruik gemaakt van ontwerpen. Wij zijn gewoon aan de slag gegaan met het project.