Adafruit Gemma
Blad 3
Servo motor

Introductie

Een servo heeft drie aansluitingen:

- Rode kabel: +
- Bruine kabel: - (ground / aarde)
- Gele kabel: data (geeft de positie van de servo door)

De kabels kunnen ook andere kleuren hebben. Bij twijfel: vraag de docent.

Benodigdheden

- Adafruit Gemma
- Servo motor
- 3 Krokodillenklemmen
- 3 kabeltjes met pinnetjes aan de uiteinden
- USB kabel en computer

Schakeling

Omdat het uiteinde van de servokabel geen pinnetjes maar gaatjes heeft, gebruik je kabels die je in deze gaatjes kunt prikken. Verbind de kabels vervolgens weer met de krokodillenklemmen aan de servo.

Bruine servo kabel (-) naar GND
Gele servo kabel (data) naar D0
Rode servo kabel (+) naar Vout

Sketch

Libraries installeren

Wanneer je Codebender of Arduino Create Web Editor gebruik kun je deze stap overslaan en onderstaande programmacode direct gebruiken.
Gebruik je de Arduino IDE, dan is het noodzakelijk dat je eerst eenmalig de SoftServo library installeert:

- Klik in het menu op Schets → Bibliotheek gebruiken → Bibliotheken beheren.
- Zoek in het filtervak naar “softservo”.
- Installeer de “Adafruit SoftServo library”.

Programmacode

Download de sketch op www.makerklas.nl (Naslagwerk – Elektronica – Adafruit Gemma).

```
/***
 * Demonstratie van een servo op een Adafruit Gemma
 */

#include <Adafruit_SoftServo.h>  // SoftwareServo (works on non PWM pins)

// Servo wordt aangesloten op D0
#define SERVOPIN 0

Adafruit_SoftServo myServo; // Maak een myServo object

void setup() {
  // Set up the interrupt that will refresh the servo for us automagically
  OCR0A = 0xAF;            // any number is OK
  TIMSK |= _BV(OCIE0A);    // Turn on the compare interrupt (below!)

  myServo.attach(SERVOPIN);  // Verbind de servo met D0 op de Gemma
  myServo.write(90);         // Tell servo to go to position per quirk
  delay(15);                 // Wacht 15ms totdat de servo in de juiste positie staat
}

void loop()  {
  int servoPos; // Deze variabele wordt gebruikt voor de positie waarin de servo staat

  // Varieer servoPos van 0 tot 180 in stappen van 1 en stel telkens de nieuwe positie in
  for (servoPos=0; servoPos<180; servoPos++) {
      myServo1.write(servoPos);
      delay(50);
  }

  // Varieer servoPos van 180 tot 0 in stappen van -1 en stel telkens de nieuwe positie in
  for (servoPos=180; servoPos>0; servoPos--) {
      myServo1.write(servoPos);
      delay(50);
  }
}

// We'll take advantage of the built in millis() timer that goes off
// to keep track of time, and refresh the servo every 20 milliseconds
volatile uint8_t counter = 0;
SIGNAL(TIMER0_COMPA_vect) {
  // this gets called every 2 milliseconds
  counter += 2;
  // every 20 milliseconds, refresh the servos!
  if (counter >= 20) {
    counter = 0;
    myServo1.refresh();
    myServo2.refresh();
  }
}
```