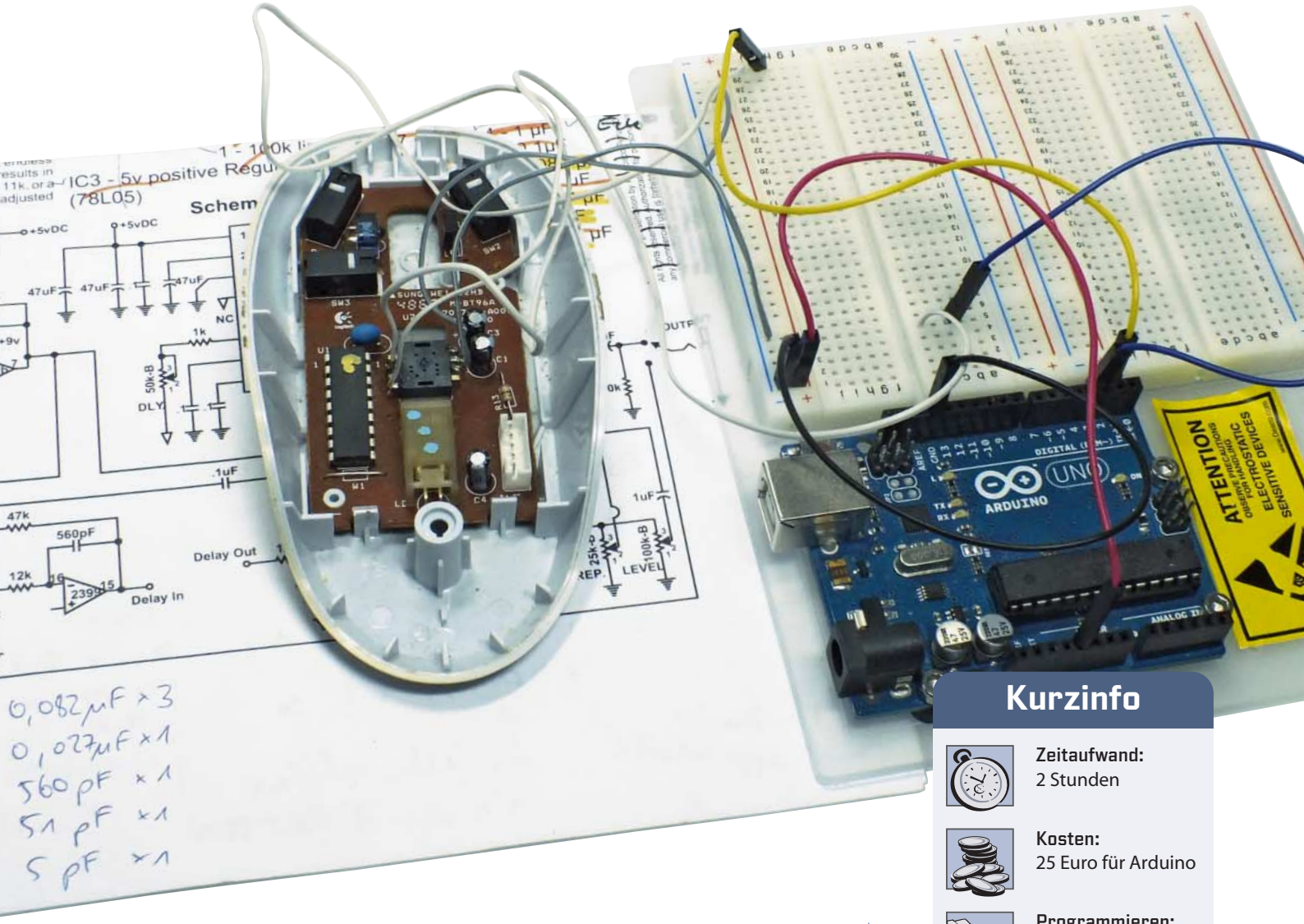
Daniel Bachfeld

Mauskamera

Die Sensoren optischer Mäuse haben spezielle Funktionen, die meist brachliegen. Mit einem Arduino kann man sie nutzbar machen und beispielsweise die Maus in eine Miniatur-Kamera verwandeln.



Kurzinfo

Zeitaufwand:
2 Stunden

Kosten:
25 Euro für Arduino

Programmieren:
Arduino, Processing

Löten:
leichte Lötarbeiten

Schwierigkeitsgrad

leicht — schwer

Zutaten

- optische Maus mit ADNS2610
- Arduino
- Breadbord und Jumperkabel, optional

© Copyright by Heise Zeitschriften Verlag

Die optische Maus hat die Kugelmaus aus ihrem Lebensraum längst verdrängt. Nun geht es in Zeiten tatschbarer Laptops, Tablets und Smartphones auch dieser Art an den Kragen. Doch statt sie zu entsorgen, lässt sich ihr Inneres für allerhand Elektronikprojekte wiederverwenden. Am interessantesten ist der optische Sensor, der aus der Mausbewegung auf der Unterlage die zurückgelegten Strecken für die x- und y-Achse berechnet. Daraus leitet sich die Bewegung des Zeigers auf grafischen Oberflächen ab.

Der Sensor ist im Prinzip eine kleine Kamera mit bespielsweise 18 × 18 Pixeln, der ein Bild von der Tischoberfläche aufnimmt. Das es unter der Maus relativ dunkel ist, erhellt eine LED die Szenerie – bei Laser-Mäusen hingegen ein Laser (wobei der nicht nur zur Erhellung beiträgt, sondern durch den Speckle-Effekt auch auf glatten Oberflächen ein erkennbares Muster erzeugt). Die Kamera nimmt über 1000 Bilder pro Sekunde auf und ermittelt über das Prinzip des optischen Flusses (siehe Link), in welche Richtung sich die Bilder verschieben.

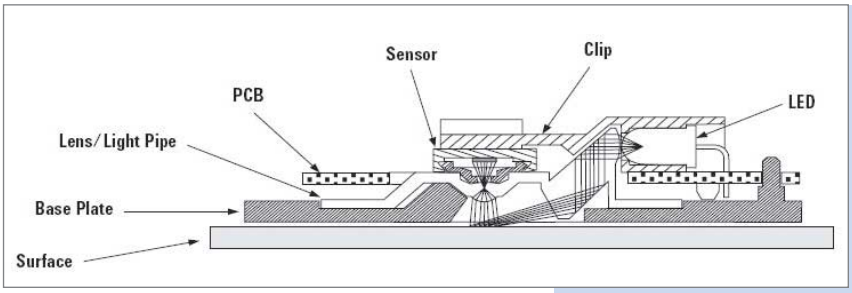
Praktischerweise lassen sich bei vielen Sensoren neben den Daten für die x- und y-Bewegungen auch die Pixeldaten selbst auslesen, sodass sich der Sensor als Minikamera einsetzen lässt. Zusammen mit einem Arduino kann man interessante Anwendungen umsetzen, beispielsweise die Maus als Miniscanner, für Positioniersysteme oder zur Beobachtung von Fadenwürmern einsetzen.

Technik

Im Test haben wir uns den Sensor-Chip ADNS2610 näher angeschaut, wie er in sehr vielen Mäusen von Logitech, Dell und anderen Herstellern verbaut ist. Er hat zwei Anschlüsse für einen Resonator zur Takterzeugung, zwei für die Versorgungsspannung (Vcc und GND) sowie zwei für eine bidirektionale synchrone Schnittstelle (SDIO und SCK). Über die Schnittstelle kommuniziert er mit einem weiteren Chip, der die Koordinaten ausliest und per USB an einen PC weiterleitet. Ein Arduino kann diese Funktion ebenfalls übernehmen und zusätzlich noch das gesamte Kamerabild zur weiteren Verarbeitung an den PC senden.

Der Chip enthält mehrere Konfigurations- und Datenregister (siehe Bild auf Seite 62). Zum Schreiben eines Bytes in ein Register erzeugt der Arduino einen Takt und legt dazu die einzelnen Bits des Adressbytes sowie des Datenbytes an den Daten-Pin des ADNS2610. Zum Auslesen eines Registers muss man zunächst die Adresse des gewünschten Registers an den Chip übermitteln, der daraufhin das Datenbyte als Bitfolge sendet – den Takt erzeugt weiterhin der Arduino. Auf diese Weise kann man etwa die Register Delta-X und Delta-Y nach einer Positionsänderung der Maus auslesen oder die 324 Graustufenwerte der einzelnen Kamera-Pixel.

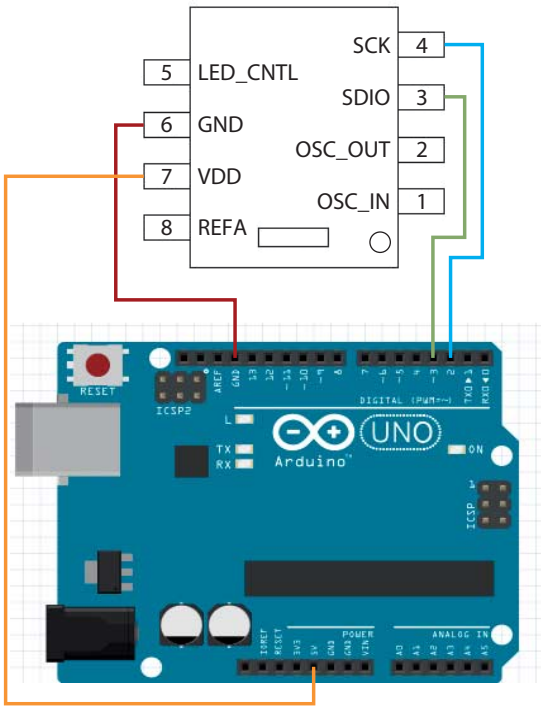
Der Codeschnipsel in Listing 1 zeigt die grundsätzliche Arbeitsweise, das vollständige Programm finden Sie zum Download unter dem Link. Mit dem Setzen des höchstwertigen Bits (|= 0x80) in der Registeradresse signalisiert der Sender einen Schreibzugriff. Die Schleife sorgt dafür, dass jedes Bit über den Pin 2 des Arduino gesendet wird. Der Schleifenzähler wird dabei nicht wie üblich in- oder dekrementiert, sondern bitweise nach rechts verschoben. Das ver-



Eine LED leuchtet die Unterlage aus. Eine Optik aus Plastik lenkt das Licht auf den Kamerasensor.

Listing 1

```
void writeRegister(byte address, byte data) {
  address |= 0x80;
  pinMode (SDIO, OUTPUT);
  for (byte i = 128; i > 0 ; i >>= 1) {
    digitalWrite (SCLK, LOW);
    digitalWrite (SDIO, (address & i) != 0 ? HIGH : LOW);
    digitalWrite (SCLK, HIGH);
  }
  for (byte i = 128; i > 0 ; i >>= 1) {
    digitalWrite (SCLK, LOW);
    digitalWrite (SDIO, (data & i) != 0 ? HIGH : LOW);
    digitalWrite (SCLK, HIGH);
  }
  delayMicroseconds(100);
}
```



Wenn der Arduino angeschlossen ist, sollte die Maus nicht mehr per USB mit einem PC verbunden sein.

© Copyright by Heise Zeitschriften Verlag

Konfigurationsregister

Register	Address	Notes
Configuration	0 × 00	Reset, Power Down, Forced Awake, etc.
Status	0 × 01	Maus wach oder standby?
Delta_Y	0 × 02	Bewegung x
Delta_X	0 × 03	Bewegung y
SQUAL	0 × 04	Measure of the number of features visible by the sensor
Maximum_Pixel	0 × 05	Fehlende
Minimum_Pixel	0 × 06	
Pixel_Sum	0 × 07	
Pixel Data	0 × 08	Bilddaten
Shutter_Upper	0 × 09	
Shutter_Lower	0 × 0A	
Inverse Product	0 × 11	Inverse Product ID

Der Maussensor hat diverse Register, über die er sich konfigurieren und auslesen lässt.

```
Listing 2

Byte res = 0;
for (byte i=128; i >0 ; i >>= 1) {
  digitalWrite (SCLK, LOW);
  digitalWrite (SCLK, HIGH);
  if( digitalRead (SDIO) == HIGH )
    res |= i;
}
```

Maske	10000000	Bit 7
Byte	10011011	
1 ←	10000000	
Maske	01000000	Bit 6
Byte	10011011	
0 ←	00000000	
⋮		
Maske	00000010	Bit 1
Byte	10011011	
1 ←	00000010	
Maske	00000001	Bit 0
Byte	10011011	
1 ←	00000001	

Der obere Wert stellt die Maske dar, um das über SDIO zu sendende Bit auszumaskieren.

einfacht die im Programm folgenden boolschen Verknüpfungen, die Langform der Operation wäre i = i > 1. Der Ausdruck (address & i) != 0 ? HIGH : LOW) ist eine verkürzte Schreibweise einer If-Then-Else-Abfrage. Wenn die Verknüpfung von address & i nicht null ergibt, ist der Wert des Ausdrucks HIGH, andernfalls LOW.

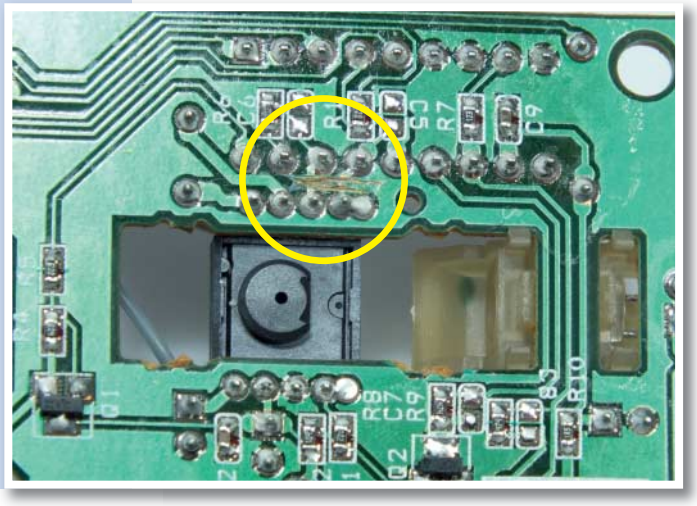
Analog zum Schreiben funktioniert das Einlesen der vom Chip zurückgesendeten Daten (Listing 2). Wieder wird der Schleifenzähler geschoben (geshiftet) statt dekremtiert. Sofern am Eingangspin ein High anliegt, wird in der Variablen res das jeweilige Bit gesetzt (|= i)

Damit der Chip ein vollständiges Bild seiner Kamera sendet, muss man irgendeinen beliebigen Wert in das Pixel-Register schreiben und anschließend die 324 Bytes bitweise einlesen. Jedes Byte enthält einen Grauwert von 0 bis 63. Die Funktion getFrame unseres Sketches liest die 324 Bytes für die Pixel ein, speichert sie in einem Array und sendet sie anschließend über die serielle Schnittstelle an den PC (Listing 3). Wichtig ist zu wissen, dass die Funktion Serial.write() die Daten als binären Wert über die serielle Leitung sendet und nicht als eine vom Anwender lesbare dargestellte Zahl.

Der Sketch ist so ausgelegt, dass man zu Testzwecken mit einem Terminalprogramm die Maus abfragen kann. Ein „d“ veranlasst den Arduino, die Binärdaten des Bildes zu senden. Die Eingabe „c“ gibt die relative Koordinatenverschiebung in X- und Y-Richtung aus.

Um das Kamerabild auf dem PC anzuzeigen, haben wir ein kleines Programm mit der Programmierumgebung Processing entwickelt, die unter Windows, Linux und Mac läuft. Der Processing-Sketch sendet regelmäßig den Befehl „d“ an den Ar-

Damit der USB-Chip dem Arduino bei der Kommunikation mit dem Pixel-Chip nicht in die Quere kommt, trennt man besser die alten Leiterbahnen auf.

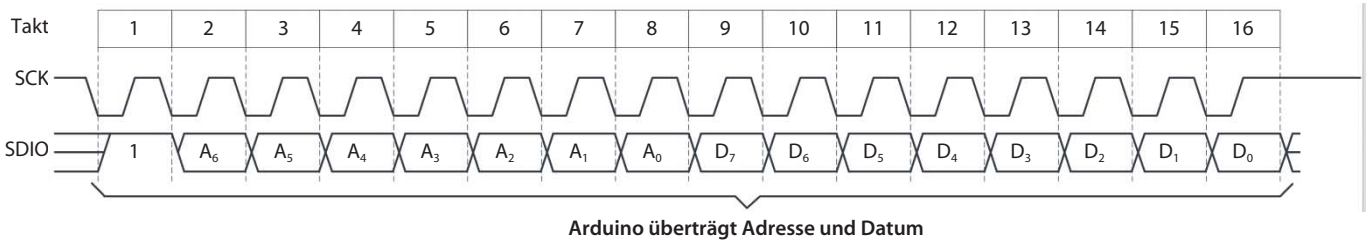


```
Listing 3

void getFrame(byte* pixel)
{

  writeRegister(REG_PIXEL_DATA, 0xAA);
  delayMicroseconds(500);
  int count = 0;

  do {
    byte data = readRegister(REG_PIXEL_DATA);
      frame[count++] = ((data & B00111111));
      delayMicroseconds(500);
    }
  while (count != 324);

  for(int i = 0; i < camHeight; i++) {
    for(int j = 0; j < camWidth; j++) {
        byte pix = (frame[(i*18) + j]);
        Serial.write(pix);
    }
  }
}
```


© Copyright by Heise Zeitschriften Verlag

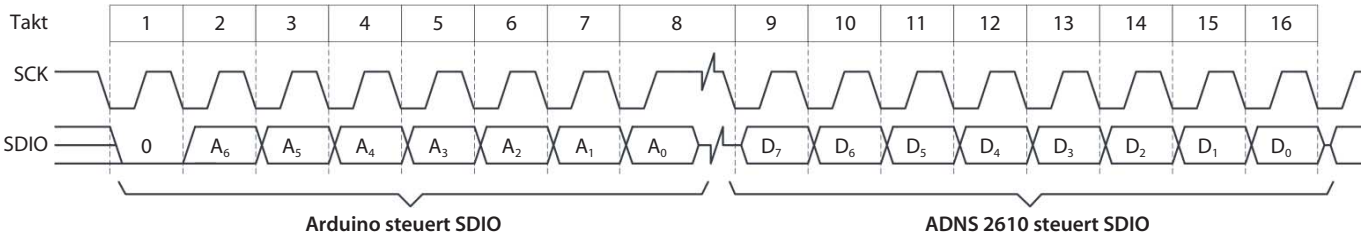
Schreibzyklus

Beim Schreiben erzeugt der Arduino sowohl den Takt als auch die Bits für Adresse und das Datum.



Lesezyklus

Zum Lesen muss das höchstwertige Bit im Adressbyte auf null gesetzt sein.



Anzeige

© Copyright by Heise Zeitschriften Verlag

Last Pixel

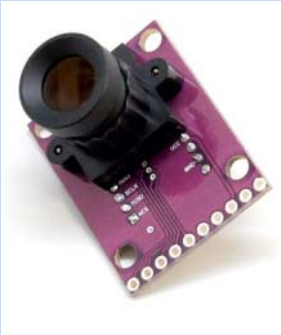
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324
17	35	53	71	89	107	125	143	161	179	197	215	233	251	269	287	305	323
16	34	52	70	88	106	124	142	160	178	196	214	232	250	268	286	304	322
15	33	51	69	87	105	123	141	159	177	195	213	231	249	267	285	303	321
14	32	50	68	86	104	122	140	158	176	194	212	230	248	266	284	302	320
13	31	49	67	85	103	121	139	157	175	193	211	229	247	265	283	301	319
12	30	48	66	84	102	120	138	156	174	192	210	228	246	264	282	300	318
11	29	47	65	83	101	119	137	155	173	191	209	227	245	263	281	299	317
10	28	46	64	82	100	118	136	154	172	190	208	226	244	262	280	298	316
9	27	45	63	81	99	117	135	153	171	189	207	225	243	261	279	297	315
8	26	44	62	80	98	116	134	152	170	188	206	224	242	260	278	296	314
7	25	43	61	79	97	115	133	151	169	187	205	223	241	259	277	295	313
6	24	42	60	78	96	114	132	150	168	186	204	222	240	258	276	294	312
5	23	41	59	77	95	113	131	149	167	185	203	221	239	257	275	293	311
4	22	40	58	76	94	112	130	148	166	184	202	220	238	256	274	292	310
3	21	39	57	75	93	111	129	147	165	183	201	219	237	255	273	291	309
2	20	38	56	74	92	110	128	146	164	182	200	218	236	254	272	290	308
1	19	37	55	73	91	109	127	145	163	181	199	217	235	253	271	289	307

First Pixel

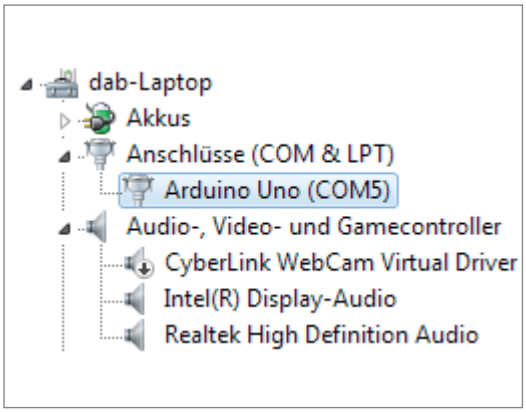
Beim Auslesen des Kamerabildes überträgt die Maus seriell 324 Bytes. Welches Byte welchen Punkt repräsentiert, zeigt die Matrix im Bild.

duino, der daraufhin 324 Bytes zurückliefert. Zur Initialisierung der seriellen Schnittstelle auf dem PC dient die Funktion:

```
void initSerial()
{
  //String portName = Serial.list()[1];
  String portName = "/dev/ttyACM0";
  port = new Serial(this, portName, rate);
  println("Using " + portName + " as serial device.");
}
```



Mit einem zusätzlichen Objektiv kann der Maussensor auch Oberflächen in der Ferne erkennen.



Nummernverwirrung: COM5 ist in diesem Beispiel die einzige verfügbare serielle Schnittstelle, an der der Arduino hängt. Im Listing muss dann Serial.list()[0]; gesetzt sein.

Listing 4

```
void draw()
{
  if( millis() >= nextFrameTime )
  {
    requestFrame();

    background(245);
    for( int x = 1; x < 18; x++ )
      {
      for (int y = 1; y < 18; y++)
        {
      fill( map(frame[(18*x)-y], 0, 63, 0, contrast) );
      rect( x * sz, y * sz, sz, sz);
        }
      }
    nextFrameTime = millis() + framePeriod;
  }
}
```

Unter Linux heißt die Schnittstelle zum Arduino /dev/ttyACM0. Unter Windows muss man den richtigen COM-Port selbst herausfinden. Je nach Anzahl der echten und virtuellen COM-Ports muss man Serial.list()[1]; anpassen und die 1 beispielsweise gegen eine 2 ersetzen. Die im Gerätemanager zu findenden COM-Port-Nummern führen dabei allerdings in die Irre: Der COM-Port 36 ist keineswegs der 36. COM-Port, wenn er in der Liste aller verfügbaren COM-Ports an zweiter Stelle steht. Unter Windows kommentiert man im Processing-Sketch die Zeile für Linux durch Voranstellen zweier Slashes aus und löscht die zwei Slashes in der darüberstehenden Zeile – und trägt die richtige Portnummer ein.

Kernstück des unter dem Link zum Download angebotenen Processing-Sketches ist die Funktion draw(). Sie zeichnet jeden Pixel der Mauskamera als Riesenpixel mit den Abmessung 22 × 22 Pixel und einem Grauwert. Der zwischen 0 und 63 liegende originale Grauwert wird der besseren Sichtbarkeit halber auf einen am PC besser erkennbaren Grau-

Sensor-Vielfalt

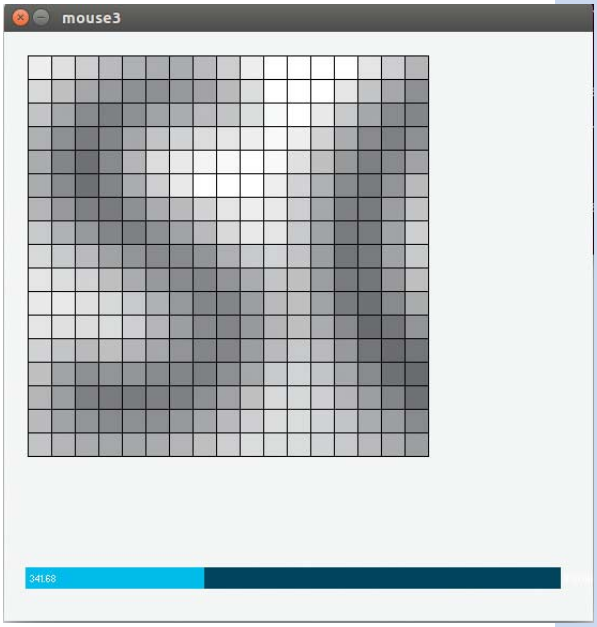
Neben dem ADNS2610 gibt es noch andere Sensoren, die alle auf dem gleichen Prinzip des „Optical Flow" beruhen. Sie unterscheiden sich jedoch in der Anzahl der Pixel und anderen Eigenschaften. Meist weichen auch die Registeradressen voneinander ab, sodass der Sketch für den ADNS2610 mit dem ebenfalls verbreiteten ADNS5020 nicht funktioniert. Anhand des Datenblatts für den jeweiligen Chip lassen sich aber der Arduino- und Processing-Sketch anpassen.

© Copyright by Heise Zeitschriften Verlag

wert übersetzt. Dazu dient der Wert contrast, der mittels eines Sliders zwischen 0 und 1023 einstellbar ist.

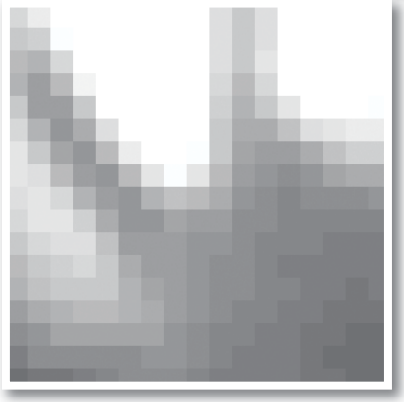
Um das Bild gemäß der Position der einzelnen Pixel auf dem PC auf der X- und Y-Achse richtig darzustellen, muss man quasi die Y-Achse auf den Kopf stellen – denn der Koordinatenursprung liegt bei Computergrafik traditionell links oben und nicht links unten. Im Koordinatensystem der mit Processing dargestellten Grafik entspricht Pixel 1 dem Punkt X=1, Y=18, Pixel 18 entspricht X=1, Y=1. Durch die zwei verschachtelten Schleifen zeichnet das Programm zunächst 18 Pixel mit X=1 und Y von 1 bis 18 mit den Grauwerten der Kamerapixel. Der Befehl map transponiert die Grauwerte aus dem Array frame[] einzeln wie oben bereits beschrieben auf einen anderen Grauwert. Mit der Anweisung fill legt Processing den Farbwert fest, mit dem Zeichen-Primitive wie Rechtecke und Kreise ausgefüllt werden. Der Befehl rect zeichnet schließlich das Rechteck (Listing 4).

Das Programm ruft das Kamerabild dreimal pro Sekunde ab und zeichnet auf dem Bildschirm die Pixel.



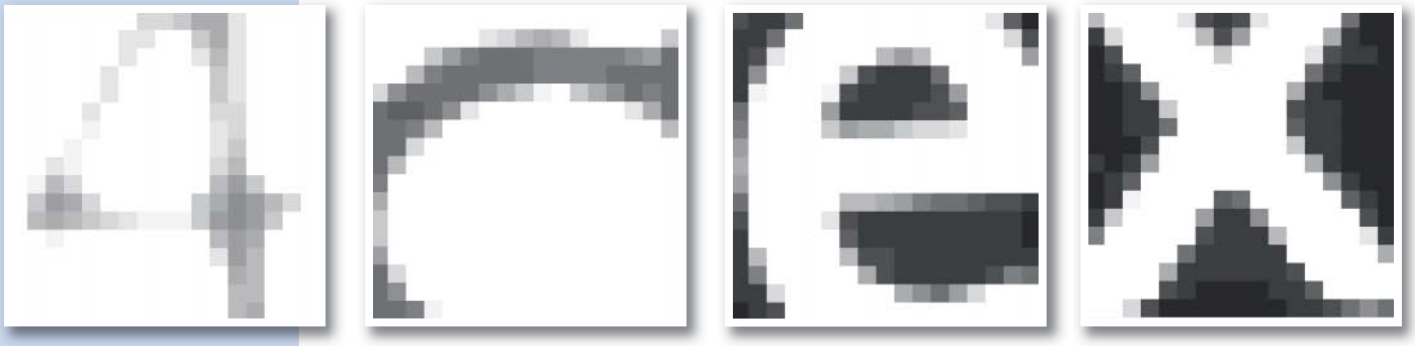
Mit dem Slider lässt sich die Helligkeit des aktuellen Kamerabildes einstellen.

Mit einen Objektiv lässt sich die Brennweite vergrößern und man kann Umrisse von Objekten erkennen.



Anzeige

© Copyright by Heise Zeitschriften Verlag



Der Sensor kann gut einzelne Buchstaben in Texten erfassen.

Praxis

Für erste Tests legt man unter den Maussensor eine dicht beschriebene Seite. Der PC sollte dann je nach Schriftgröße einzelne Buchstaben oder Fragmente davon anzeigen. Mit dem Slider lässt sich die Helligkeit respektive der Kontrast verändern. Bewegt man die Maus über die Unterlage, kann man größere Strukturen erfassen. Grundsätzlich ließe sich der Maussensor als Mini-Scanner verwenden, der aus den Einzelbildern und deren Positionen eine größere Vorlage wiederherstellt. Eine Implementierung für Windows gibt es auf der Seite Spritemods (siehe Link).

Daneben gibt es Ansätze, den Sensor zur Beobachtung von Mikroorganismen und zur Stabilisierung von Multikoptern einzusetzen (siehe Link). Bei den Mikroorganismen kann man beispielsweise die Bewegungsgeschwindigkeit messen. Bei den Quadrokoptern kann der nach unten gerichtete Sensor das Driften in eine bestimmte Richtung messen und der Flightcontroller steuert dementsprechend gegen.

Da der Sensor jedoch eher für Nahaufnahmen ausgelegt ist, muss man eine zusätzlich Optik zur Fokussierung des Bildes auf den Sensor verwenden. 3D-Robotics bietet einen fertigen Sensor (ADNS3080) für solche Zwecke an, der mit dem Flightcontroller ArduPilot kompatibel ist.

Unsere Versuche mit dem Objektiv einer ausgemusterten Webcam brachten nur bescheidene Ergebnisse, wie die Bilder auf Seite 65 zeigen. Immerhin sind noch die Umrisse einer Hand zu sehen und es lässt sich unterscheiden, ob sie vier oder zwei Finger zeigt. Das lässt sich mit Geduld und weiteren Versuchen sicherlich optimieren. Finden Sie noch weitere nützliche Anwendungen? Dann schicken Sie uns Ihr Ergebnis an hacks@heise.de.

Links und Foren
www.ct.de/ch1404060

Der Aprilscherz

Die ersten Versuche mit der Mauskamera begannen wir bereits im März dieses Jahres. Zu diesem Zeitpunkt tüftelte die c't-Redaktion gerade an Ideen für den Aprilscherz, der jedes Jahr in Form eines subversiven Artikels ins Heft kommt. Da das Kamerabild auf manchen Unterlagen an einen QR-Code erinnerte, kamen wir auf die Idee zur Geschichte einer „trojanischen Maus", die so lange inaktiv bleibt, bis sie über einen winzigen QR-Code auf einem präparierten Mauspad geschoben wird. Ab da sollte sie sich als Tastatur am PC anmelden und Befehle zum Nachladen weiteren Schadcodes senden respektive tippen.

In der Tat gelang es uns in Versuchen, so kleine QR-Codes mit dem Laserdrucker herzustellen, die nur noch mit der Lupe oder der Maus erkennbar waren. Um damit etwas in der Maus auszulösen, hätte man jedoch die Firmware der Maus manipulieren müssen – was unseres Wissens nachträglich nicht möglich ist. Immerhin fielen viele c't-Leser auf den als Sicherheits-Report getarnten Artikel „Im Auge der Maus" in Ausgabe 8/14 herein, darunter sogar Datenschutzbeauftragte aus Unternehmen. Wenige erkannten den Scherz als solchen – insgesamt erreichten uns über 80 Mails. (dab)

© Copyright by Heise Zeitschriften Verlag