Robot programming with Arduino (Part 2)

In the first Part of the robotics tutorials, we have learned to communicate with the ROB0081 and created a first program. Now it comes to the basic movement and the detec
same a note: In the field of robotics, the individual challenges in most cases resolve in different ways. So it is always the creativity of the developer asked. For this reason, th
are certainly not the only way to your destination.

2 steps forward, two steps back

This not the robot does the first road test of the dust, let him only before and then let go back in reverse gear back for a short distance. The motors are controlled ROB0081
pins, the digital pins 4 to 7 These connectors are not identical to the pins of the microcontroller, but they are managed by the Arduino libraries. Microcontrollers that are Ardui
addressed with the same name, although the controller ports are perhaps others.

The functions of the four relevant pins are:
Pin 4: Direction of the left motor
Pin 5: Control of the left motor
Pin 6: Control of the right motor
Pin 7: direction of the left motor

Orig

Teil d
dem
erstell

Contri

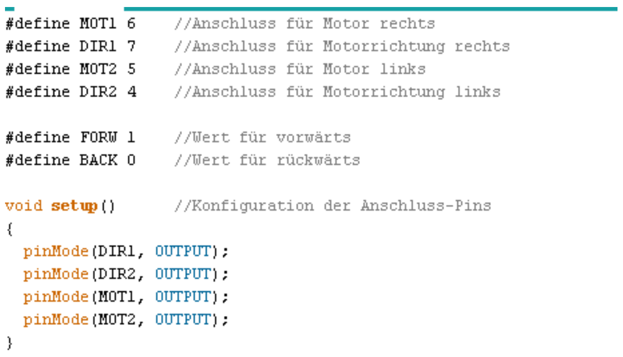
To address these pins, we need two functions:

- DigitalWrite (PIN DIRECTION)
PIN is the respective port is and DIRECTION for either a "1" for forward or a "0" for backwards is set.

- Analog write (PIN VALUE)
This function returns a number between 0 and 255 is used for VALUE (0 = stop, 255 = maximum speed

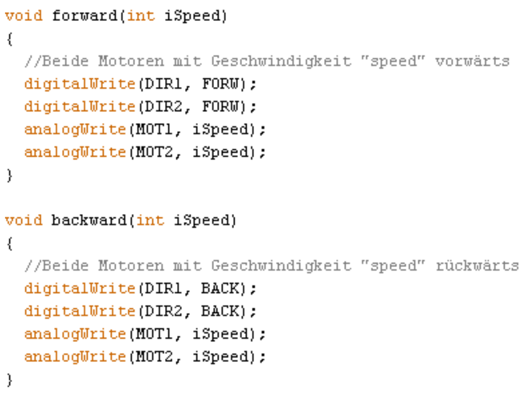
Well I have written above, there are digital pins, so how we can output a value of 255 to it? In analog write () is an Arduino function that does nothing to generate a square wa
of about 490 Hz. With VALUE simply determine the duty cycle (duty cycle). At 0, the signal at 255 is constantly out constantly and at a value of 128, it is during a period in equ
the rms value of the voltage of the area of the rectangle corresponds, can the speed of a DC motor control so directly.

Now let's create the program in the Arduino development environment:



```
#define MOT1 6    //Anschluss für Motor rechts
#define DIR1 7    //Anschluss für Motorrichtung rechts
#define MOT2 5    //Anschluss für Motor links
#define DIR2 4    //Anschluss für Motorrichtung links

#define FORW 1    //Wert für vorwärts
#define BACK 0    //Wert für rückwärts

void setup()      //Konfiguration der Anschluss-Pins
{
  pinMode(DIR1, OUTPUT);
  pinMode(DIR2, OUTPUT);
  pinMode(MOT1, OUTPUT);
  pinMode(MOT2, OUTPUT);
}
```

In the upper part first be carried back definitions, so we do not need to work with numbers with meaningful words and. DIR is the abbreviation for the directional pins and MO
connection of the respective motor control.
In the setup () function, the four pins are configured as outputs. This completes the configuration would already be done.
To make the whole thing a little clear, we write for the commands "go forward", "go backwards", "turn right" and "left turn" your own functions.



```
void forward(int iSpeed)
{
  //Beide Motoren mit Geschwindigkeit "speed" vorwärts
  digitalWrite(DIR1, FORW);
  digitalWrite(DIR2, FORW);
  analogWrite(MOT1, iSpeed);
  analogWrite(MOT2, iSpeed);
}

void backward(int iSpeed)
{
  //Beide Motoren mit Geschwindigkeit "speed" rückwärts
  digitalWrite(DIR1, BACK);
  digitalWrite(DIR2, BACK);
  analogWrite(MOT1, iSpeed);
  analogWrite(MOT2, iSpeed);
}
```

We can function forward () or backward () the parameter iSpeed passed, the values of 0 to 255 can take and determines the speed.

```
void turn_right()
{
  digitalWrite(DIR1, BACK);
  analogWrite(MOT1, 255);
  digitalWrite(DIR2, FORW);
  analogWrite(MOT2, 255);
}

void turn_left()
{
  digitalWrite(DIR1, FORW);
  analogWrite(MOT1, 255);
  digitalWrite(DIR2, BACK);
  analogWrite(MOT2, 255);
}
```

The functions turn_right () and turn_left () are virtually the same design as forward () and backward (), but there is no value for the velocity is passed and the motors rotate in

In the main program, the loop (), the individual functions are now selected in sequence:

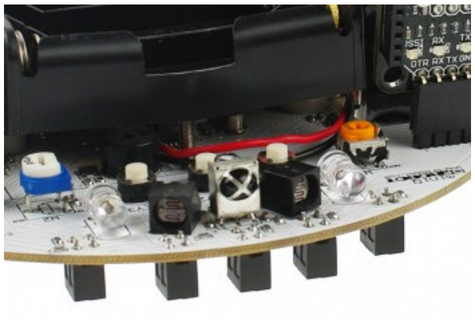
```
void loop()
{
  forward(255);        //vorwärts mit vollem Tempo
  delay(1000);         //1 Sekunde lang fahren
  forward(0);          //Motor stoppen

  turn_left();         //1 Sekunde lang links drehen
  delay(1000);
  forward(0);

  backward(128);       //rückwarts mit halbem Tempo
  delay(1000);         //2 Sekunden lang fahren
  forward(0);

  turn_right();        //1 Sekunde lang rechts drehen
  delay(1000);
  backward(0);
}
```

After compiling and uploading the robot behaves as desired. It runs at full speed 1 second after the front, turns a second after the right, go half-speed turn back 1 second and

That's for starters quite nice, but we need either a very large room or we need to look idly with how our little robot caused a crash after another. But we'll break him in the nex

Crash avoidance with infrared transceivers

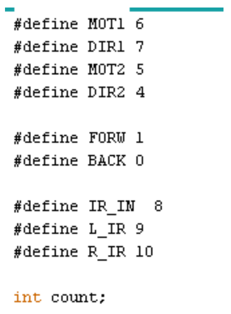
Collision avoidance is one of the standard functions in robotics. There are a variety of technical procedures in order to meet this challenge. A particularly efficient and secure
transceivers, that is, a unit of the transmitter and receiver. The advantage is that hardly comes to be susceptible to ambient radiation, because the transmitter and receiver c



The small silver sensor on the board in the middle is the IR receiver, the two transparent LEDs are the sender.

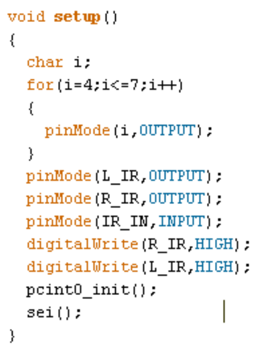
But the ROB0081 should not only recognize whether it represents an obstacle in the way he should also find out where it is located. , the principle that we use for this is ver
The transmitters produce several short pulses with a frequency of about 40 kHz. The robot is in a free field, the infrared light is hardly reflected, since the scattering of the inf
Take it now approaches an object, hit the receiver pulses. The number of the received pulses is a measure of the distance to the object. And depending on whether the puls
channel is received, knows our program, where the object is located.

Now we just have to determine if this theory works in practice and the hardware of the ROB0081 they can actually implement. We develop our program and you will be ama
Arduino fails.



```
#define MOT1 6
#define DIR1 7
#define MOT2 5
#define DIR2 4

#define FORW 1
#define BACK 0

#define IR_IN  8
#define L_IR 9
#define R_IR 10

int count;
```

We all know the upper part, in principle, already from the first example. There symbolic constants are defined only. In addition, the connector pins for the infrared receiver LR
infrared transmitter L_IR (pin 9) and R_IR (pin 10). In addition, an integer variable "count" is not defined, we need to count the pulses.

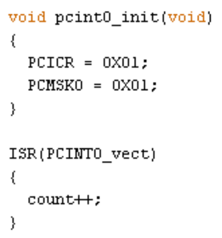


```
void setup()
{
  char i;
  for(i=4;i<=7;i++)
  {
    pinMode(i,OUTPUT);
  }
  pinMode(L_IR,OUTPUT);
  pinMode(R_IR,OUTPUT);
  pinMode(IR_IN,INPUT);
  digitalWrite(R_IR,HIGH);
  digitalWrite(L_IR,HIGH);
  pcint0_init();
  sei();
}
```

Right after that is done the setup that turns out a bit more extensive. To configure the pins 4-7 as outputs, we use, for a change a for loop. Then the connections for the IR tra
configured as outputs and the connection for the IR receiver as an input.

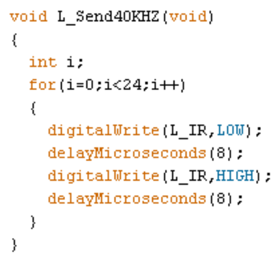
The IR LEDs are an adjustable resistor to +5 V. They are therefore turned on when the pin terminal is pulled "low." This configuration is frequently used, because most micro
higher driving power, when the current flowing in the connection as in the opposite case. For this reason we need the two digitalWrite () commands that set the ports to "high

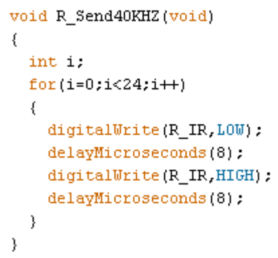
The next two instructions are required for the interrupt. We are in fact among the received pulses to the IR receiver on interrupts. Each time a pulse is received, it triggers an
our counter variable "count". For this is even more.



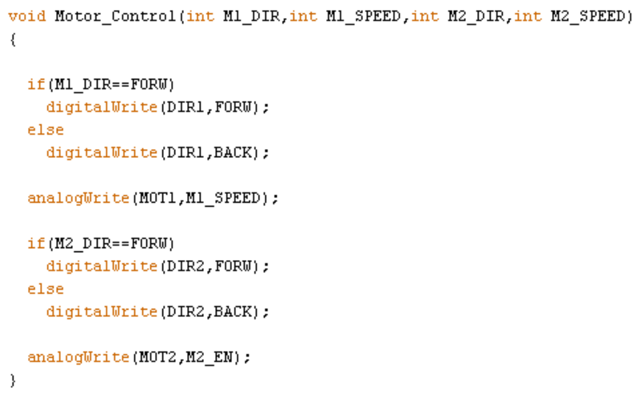
```
void pcint0_init(void)
{
  PCICR = 0X01;
  PCMSK0 = 0X01;
}

ISR(PCINT0_vect)
{
  count++;
}
```

The "pcint0_init ()" describes the interrupt configuration registers and the interrupt PCICR make PCMSK0 register. Thus, the interrupt is enabled for the connection pin of the
function of an interrupt service routine (ISR), which is a program that is executed at every interrupt (in this case a PCINT0). The only thing this does is ISR incrementing the v

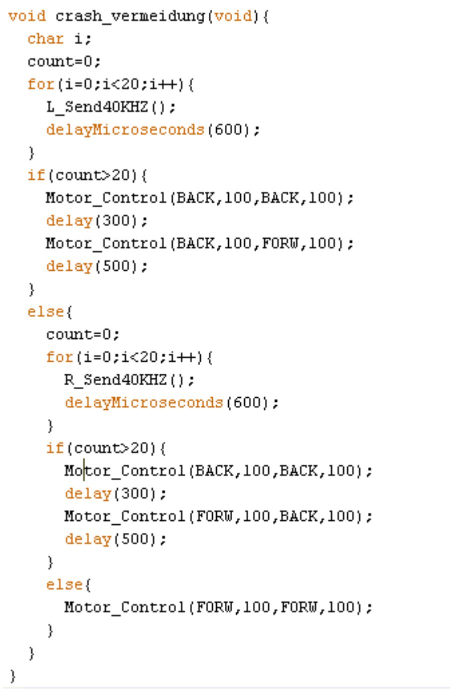

```
void L_Send40KHZ(void)
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(L_IR,LOW);
    delayMicroseconds(8);
    digitalWrite(L_IR,HIGH);
    delayMicroseconds(8);
  }
}
```


```
void R_Send40KHZ(void)
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(R_IR,LOW);
    delayMicroseconds(8);
    digitalWrite(R_IR,HIGH);
    delayMicroseconds(8);
  }
}
```

The following two functions generate the 40 kHz transmit signal for the left and right IR LED. To the pin or R_IR L_IR 24 times at an interval of 6 microseconds and off.

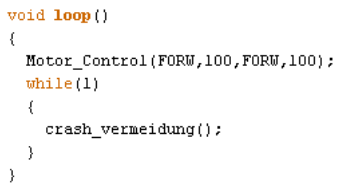

```
void Motor_Control(int M1_DIR,int M1_SPEED,int M2_DIR,int M2_SPEED)
{

  if(M1_DIR==FORW)
    digitalWrite(DIR1,FORW);
  else
    digitalWrite(DIR1,BACK);

  analogWrite(MOT1,M1_SPEED);

  if(M2_DIR==FORW)
    digitalWrite(DIR2,FORW);
  else
    digitalWrite(DIR2,BACK);

  analogWrite(MOT2,M2_EN);
}
```

The motor control we have something different structure, as in the last example. This time, four parameters are passed: direction and speed for each of the two motors. Oth
should represent nothing new for you.


```
void crash_vermeidung(void){
  char i;
  count=0;
  for(i=0;i<20;i++){
    L_Send40KHZ();
    delayMicroseconds(600);
  }
  if(count>20){
    Motor_Control(BACK,100,BACK,100);
    delay(300);
    Motor_Control(BACK,100,FORW,100);
    delay(500);
  }
  else{
    count=0;
    for(i=0;i<20;i++){
      R_Send40KHZ();
      delayMicroseconds(600);
    }
    if(count>20){
      Motor_Control(BACK,100,BACK,100);
      delay(300);
      Motor_Control(FORW,100,BACK,100);
      delay(500);
    }
    else{
      Motor_Control(FORW,100,FORW,100);
    }
  }
}
```

Now it is again exciting because we actually create the routine for crash avoidance, which we call as such. In the first for loop 20 pulses for the left IR LED are generated bet
of 600 microseconds. If an object in front of the left IR LED are, the IR light would be reflected and the IR receivers receive the pulses. The ISR would be executed and the Za
incremented. Since an IR pulse from 24 signal changes is, "count" can therefore assume a maximum value of 480 20 x 24. For the detection of an object, it is sufficient us if

only one signal is detected. If "count" takes on a value greater than 20, we define a collision. The robot moves backwards for 300 ms and then turns also 300 ms to the right
detected on the left. The rest of this function does the same for the other IR transmitter.



```
void loop()
{
  Motor_Control(FORW,100,FORW,100);
  while(1)
  {
    crash_vermeidung();
  }
}
```

Now only missing the main program loop "loop ()", which is nothing other than making the robot first forward at low speed to send off and then in the "crash_vermeidung" bra
even.

After compiling and uploading, we can now finally turn the robot without having to make about our furniture or the ROB0081 worry.

For more information and orders of the
ROB0081 MiniQ robot 2WD
simply click on the following link: DFRobot ROB0081
NEW **experimental robot RoverRom-*B3000***

They want to be Arduino professional? Then our video microcontroller course MC1 just the thing for you!

If you have questions about, please contact us.
We are happy to help!

Your team of Böcker system electronics

Note: Our Tutorial offer will be extended at irregular intervals. If you want to be informed immediately of any new publication, please sign up **here** in our notification list "Tut
unsubscribe in your account or by e-mail this courtesy at all times.