Projektstart: 20.02.2011

Hardware Aufbau und Inbetriebnahme

Lötarbeiten durchgeführt → 10 Stunden

Nacharbeiten Hardware Plattform → 3 Stunden

– 2h Zahnräder im Getriebekasten ecken, ausfeilen hat geholfen
– 1h Testen der Standardkomponenten

Hinzufügen der Extrakomponenten → 17 Stunden

7h SRF-08 Modul an I2C Bus anbinden

– Lesen Datenblatt S166 – 194
– Lesen Datenblatt SRF-08 Modul
– Implementierung im Code mithilfe des Codebeispiels auf Seite 179 und der i2cmaster.h/.c von Peter Fleury http://jump.to/fleury (Grundsätzlich ist hier nur das Prinzip vom Datenblatt bereits umgesetz)
– 2 Pull-up Widerstände sind für SCL und SDA nötig!

```
int testi2c() {
  int dist = 0;
  unsigned char tmp = 1;
  unsigned char tmp2 = 1;

  /* fire Supersonic Modul on Adress 0xE0 */
  i2c_start_wait(0xE0+I2C_WRITE); // set Bus on Adress and Write Mode
  i2c_write(0x00); // set Register to write/read
  i2c_write(0x51); // set measurement in cm and fire the supersonic pulse
  i2c_stop(); // release bus
```

```
delay_ms(100); // waiting for echos

do {

//waiting for module to be ready again (return value == 255 when busy)
i2c_start_wait(0xE0+I2C_WRITE);
i2c_write(0x00);
i2c_rep_start(0xE0+I2C_READ);
tmp = i2c_readNak();
i2c_stop();

}while(tmp == 255);

//get high byte
i2c_start_wait(0xE0+I2C_WRITE);
i2c_write(0x02);
i2c_rep_start(0xE0+I2C_READ);
tmp = i2c_readNak();
i2c_stop();

//get low byte
i2c_start_wait(0xE0+I2C_WRITE);
i2c_write(0x03);
i2c_rep_start(0xE0+I2C_READ);
tmp2 = i2c_readNak();
```

```
 i2c_stop();

 dist = tmp*255 + tmp2; // put together low byte and high byte into returnvalue

 return dist;
 }
```

2h Servo Software PWM

– kein Hardware PWM mehr frei da von den Motorbrücken bereits belegt
– PWM muss nicht in eine ISR da der Servo zwar drehen muss, allerdings kein Gewicht halten muss

```
void servo(double winkel)
{
  unsigned int count=0;
  double middle = 1.585; //pulse length middle

  do
  {
    count++;

   if(middle)
   {
     PORTC |= (1 << PC2); // set servo control line
     delay_ms(middle + winkel);
   }
   PORTC &= ~(1 << PC2); // reset servo control line
```

```
    delay_ms(20-(middle+winkel));
  }
  while (count<30); // pulse 30 times until the servo reaches the desired position
}
```

– 7h Bluetooth Modul
– Lesen Datenblatt S137 - 165
– Spannungsregler 3.3V (LD33V)
– Vorwiderstand für TX und RX Leitung da Spannungsdifferenz (2,2k geschätzt)
– Die Standardlib der Nibobee hat nicht funktioniert, deswegen selbst implementiert

```
#define F_CPU 15000000UL
#define BAUD 19200UL
#define UBRR_VAL ((F_CPU+BAUD*8)/(BAUD*16)-1) // Universal Baud Rate Register calculation
//Initialize UART
void uart_init(void)
{
  UCSRB |= (1<<TXEN);  // UART TX einschalten
  UCSRB |= (1<<RXEN);  // UART RX einschalten
  UCSRC |= (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);  // Asynchron 8N1

  UBRRH = UBRR_VAL >> 8; // SetBaudRate High Byte
  UBRRL = UBRR_VAL & 0xFF; // SetBaudRate Low Byte
}
```

```
//Blocking getc from uart
uint8_t uart_getc(void)
{
    while (!(UCSRA & (1<<RXC)))   // warten bis Zeichen verfuegbar (RXC → receive complete flag)
        ;
    return UDR;                 // Zeichen aus UDR an Aufrufer zurueckgeben
}


//Non-blocking getc from uart
uint8_t uart_getc_nb(void)
{
    if (!(UCSRA & (1<<RXC))) {  // wenn Zeichen verfügbar (RXC → receive complete flag)
            return 0;
    } else {
        return UDR;
        }               // Zeichen aus UDR an Aufrufer zurueckgeben
}
// Wrapper um uart_getc() für ganzen String
void uart_gets( char* Buffer, uint8_t MaxLen )
{
  uint8_t NextChar;
  uint8_t StringLen = 0;

  NextChar = uart_getc();        // Warte auf und empfange das naechste Zeichen
```

```
  // Sammle solange Zeichen, bis:
  // entweder das String Ende Zeichen kam
  // oder das aufnehmende Array voll ist
  while( NextChar != '\n' && StringLen < MaxLen - 1 ) {
    *Buffer++ = NextChar;
    StringLen++;
    NextChar = uart_getc();
  }

                          // Noch ein '\0' anhaengen um einen Standard
                          // C-String daraus zu machen
  *Buffer = '\0';
}

//Write one char to uart
void writeChar(uint8_t data )
{
 while ( !( UCSRA & (1<<UDRE)) ); // wait for empty transmit buffer
 UDR = data; // Put data inot buffer, sends the data
}

// Wrapper around writeChar for Strings
void writeString(char *string)
{
  while(*string)
```

```
        writeChar(*string++);
}
```

1h zwei superhelle Leds als Scheinwerfer (um auf Dunkelheit über das SRF-08 Modul reagieren zu können)

– Vorwiderstände LEDs 47Ohm

LEDs ON: `PORTC |= (1 << PC2) | (1 << PC3);`
LEDs OFF: `PORTC &= ~(1 << PC2);`
`PORTC &= ~(1 << PC3)`

Anwendungen:

8h Erste Versuche einen Linienfolgealgorithmus als Finite State Maschine zu implementieren (noch ohne Ultraschallmodul, nur mit Fühlern)

```
while(1)  {
    left = line_get(LINE_L);
    middle = line_get(LINE_C);
    right = line_get(LINE_R);

    sensL = sens_getLeft();
    sensR = sens_getRight();

    if(sensL) presentstate = left_collision;
```

```
if(sensR) presentstate = right_collision;

if(presentstate == online || presentstate == left_offset || presentstate == right_offset) { // Auf Linie oder steuernd und auf Linie
    if (left > LINELOSTTHRESHOLD && middle < LINELOSTTHRESHOLD && right > LINELOSTTHRESHOLD) { presentstate = online; } // Strich direkt in der Mitte
    else if(left > LINELOSTTHRESHOLD && middle > LINELOSTTHRESHOLD) { presentstate = left_offline; } //Strich recht draussen
    else if (right > LINELOSTTHRESHOLD && middle > LINELOSTTHRESHOLD) { presentstate = right_offline; } // Strich links draussen
    else if (right > LINELOSTTHRESHOLD && middle < LINELOSTTHRESHOLD && left < LINELOSTTHRESHOLD) { presentstate = right_offset; } //Strich driftet links ab
    else if (left > LINELOSTTHRESHOLD && middle < LINELOSTTHRESHOLD && right < LINELOSTTHRESHOLD) { presentstate = left_offset; } //Strich driftet rechts ab
    else { presentstate = online; }
} else if (presentstate == left_offline) {
    if (middle < LINEFOUNDTHRESHOLD) { presentstate = online; } //Linie wieder gefunden
    else { presentstate = left_offline; }
} else if (presentstate == right_offline) {
    if (middle < LINEFOUNDTHRESHOLD) { presentstate = online; } //Linie wieder gefunden
    else { presentstate = right_offline; }
}

switch(presentstate) {
    case online:        motpwm_setLeft(SPEED);
                        motpwm_setRight(SPEED);
                        break;
    case left_offset:   motpwm_setLeft(SPEED);
                        motpwm_setRight(SPEED-150);
                        break;
    case right_offset:  motpwm_setLeft(SPEED-150);
                        motpwm_setRight(SPEED);
```

```
                                    break;
        case right_offline:     motpwm_setLeft(SPEED_LOW);
                                    motpwm_setRight(SPEED);
                                    break;
        case left_offline:      motpwm_setLeft(SPEED);
                                    motpwm_setRight(SPEED_LOW);
                                    break;
        case left_collision:    motpwm_setLeft(0);
                                        motpwm_setRight(0);
                                        avoid_left();
                                        presentstate = online;
                                        break;
        case right_collision:   motpwm_setLeft(0);
                                        motpwm_setRight(0);
                                        avoid_right();
                                        presentstate = online;
                                        break;
    }
}
```

3h Fernsteuerung über das BTM Modul

```
    while(1) {
      c = uart_getc();

       if (c == 'w') {
```

```
            move_odo(SPEEDL,SPEEDR,5,5);
        } else if (c == 'd') {
            move_odo(SPEEDL,-SPEEDR,2,-2);
        } else if (c == 'a') {
            move_odo(-SPEEDL,SPEEDR,-2,2);
        } else if (c == 's') {
            move_odo(-SPEEDL,-SPEEDR,-5,-5);
        } else if (c == 'e') {
            search_widest_distance();
        } else if (c == 'r') {
            i = testi2c();
            writeString("distance: ");
            itoa(i,buf,10);
            writeString(buf);
            i = readi2c_light();
            itoa(i,buf,10);
            writeString("\nlight: ");
            writeString(buf);
            writeString("\n");
        } else if (c == 'f') {
            show_status();
        } else if (c == 't') {
            inside_supersonic();
        } else if (c =='g') {
            drive_around_supersonic();
        }
}
```

21.03.2011