Der neue RP6-ESP

Hier soll der RP6 (genauer der RP6v2-Base) mit einem ESP kombiniert werden.

Hardware

- RP6v2
- Bumper-Board der MultiIO, mit zwei Bumpern und zwei Sharp-IR-Abstandssensoren sowie zugehörigem Transistor
- LED-Platinen der ArduiIO
- auf der Basis werden zwei BC847 aufgebaut, um die Scheinwerfer über die Status-LEDs der RP6-Base zu schalten; die Blinker werden direkt an die Status-LEDs der RP6-Base angeschlossen. Scheinwerfer Front/Heck innen (lila Kabel, IO2), Scheinwerfer Front/Heck außen (grünes Kabel, IO5), Blinker links (oranges Kabel, IO4) und Blinker rechts (gelbes Kabel, IO1).
- Die Bumper (ON-L und ON-R) sollen noch an die IO1 (rechts) und IO5 (links) angeschlossen werden. Außerdem der Transistor vor den Sharp-Abstandssensoren auf LIO1 (dort ist auch der rechte vordere Bumper angeschlossen). Und die beiden Sharp-Abstandssensoren kommen auf ADC0 und ADC1.
- ESP32 mit 4fach-Pegelwandler auf Erweiterung. Der ESP kommuniziert via UART (Rx2/Tx2) mit dem RP6. Die Baudrate beträgt 38.400.
- Der ESP ist wie folgt angeschlossen: Pin Rx2 (GPIO16) via Pegelwandler auf Tx des RP6; Pin Rx2 (GPIO17) via Pegelwandler auf Tx des RP6. Pin D21 auf ST2 des RP6. Pin D19 via Pegelwandler auf ST1 des RP6.
- ST1/ST2 des RP6 starten/resetten den RP6: zieht man ST1 auf +5V (HIGH) startet der RP6. Zieht man den ST2 auf GND (LOW) so resettet er.

Firmware RP6

Wie beschrieben kommt im Wesentlichen die Firmware für den RP6-Base von FabianE.'s Remotrol zum Einsatz. Hinzugefügt wurde in der RP6RobotBaseLib.c der Absatz zu den BackBumpers:

```
//*********NEW**********
// from: nil.at, RP6-Forum


uint8_t getBackBumperLeft(void)
```

```
{
   PORTB &= ~SL4; □□□□//Schalte StatusLED4 (liegt auf PortB) ab
   DDRB &= ~SL4; □□□□//Schalte StatusLED4 auf Eingang
   nop(); □□□□□□//warte kurz
   uint8_t tmp = PINB & SL4; □//Lege den Wert von StatusLED4 auf "tmp"
   if(statusLEDs.LED4) □□□//wenn StatusLED4 vorher schon an war
   {
      DDRB |= SL4; □□□□//Schalte StatusLED4 auf Ausgang
      PORTB |= SL4; □□□//und schalte StatusLED4 wieder an
   }
   return tmp;
}

uint8_t getBackBumperRight(void)
{
   PORTC &= ~SL1; □□□□//Schalte StatusLED1 (liegt auf PortC) ab
   DDRC &= ~SL1; □□□□//Schalte StatusLED1 auf Eingang
   nop(); □□□□□□//warte kurz
   uint8_t tmp = PINC & SL1; □//Lege den Wert von StatusLED1 auf "tmp"
   if(statusLEDs.LED1) □□□//wenn StatusLED1 vorher schon an war
   {
      DDRC |= SL1; □□□□//Schalte StatusLED1 auf Ausgang
      PORTC |= SL1; □□□//und schalte StatusLED1 wieder an
   }
   return tmp;
}
```

Entsprechend wurde auch der `BUMPERS_stateChanged_DUMMY(void)` angepasst und die RP6RobotBaseLib.h.

Software ESP32

Der auf dem ESP32 laufende Code findet sich am Ende dieses Artikels. Die Software des ESP32 wurde zunächst mit RemoteXY gemacht. RemoteXY liefert ein schönes GUI für das Smartphone, jedoch mit allerlei Einschränkungen (nur gewisse Bausteine; keine Einbindung eines Kamerabildes; bei mehr als fünf Bausteinen wird es mit 12,99 Euro kostenpflichtig!). Daher wurde doch wieder auf eigenen Code zurückgegriffen. Nun dient ein M5stickC mit JoyC-Joysticks als Remote-Controller sowie zur Anzeige der Sensordaten. Auf dem ESP32 des RP6 läuft ebenfalls eigener Code, welcher im Wesentlichen die Sensordaten via ESPnow an den M5stickC schickt und vom M5stickC Befehle

erwartet. Der Code wurde von Fabian geschrieben.

Library für den RP6 mit UART2

Es gibt eine Bibliothek RP6_UART2_LIB.h, welche es dem ESP32 ermöglicht, via UART2 mit dem RP6 zu kommunizieren.

Änderungen und Erweiterungen

Einiges mehr soll noch kommen, hier ein paar Ideen/Pläne:

Hardware

- ☐ der ESP32 soll auf eine richtige Erweiterungsplatine, mit Pegelwandler, I2C- und UART-Zugang und USRBUS/XBUS
- ☐ Linienfolgermodul der RP6-MultiIO mit Servo, um es hoch- und herunterzufahren
- ☐ Akku mit Ladeeinheit
- ☐ Ultraschall-Abstandssensor mit Drehturm und Servo
- ☐ evtl. ESP32-CAM mit in den Drehturm
- ☐ evtl. Scheinwerfer mit in den Drehturm
- ☐ Gehäuse
- ☐ OLED-Display
- ☐ IMU-BreakoutBoard

Firmware RP6

Weitere Änderungen sollen sein: Wenn das ACS (Anti-Collision-System) an ist, soll automatisch beim Rückwärtsfahren das vordere ACS de- und das hintere (die beiden IR-Sharp-Sensoren) aktiviert werden; beim Vorwärtsfahren entsprechend das hintere de- und das vordere aktiviert. Bei einer Drehung auf der Stelle sollen beide aktiv sein.

Außerdem soll bei Kurven der jeweilige Blinker blinken und bei Drehungen auf der Stelle sowie bei Zusammenstößen soll der Warnblinker an gehen.

Software ESP32

- ☑ Reset des RP6 (dadurch Stop) nach Verbindungsabbruch
- ☑ Anzeige diverser Sensoren auf der Fernsteuerung (M5stickC)

- [ ] Einbindung des Kamerabildes?
- [ ] automatisches ACS, wodurch etwa bei Gefahr automatisch angehalten wird
- [ ] Routinen (Lichtfolger, Abstandhalter, Labyrinth, evtl. auch mit AI-Komponenten wie HIER)

Code auf dem ESP32

diesen gibt's demnächst auf der Github-Seite als Beispielcode zur UART-Library RP6_UART2_LIB.h.

Revision #12
Created 19 November 2022 17:32:17 by Fabian
Updated 19 March 2024 13:42:26 by Fabian