24.1.5.3 Projekt Temperaturmessung

In diesem Projekt wird ein Temperatur-Wert ermittelt und angezeigt. Computer und die Platine sind über ihre seriellen Schnittstellen miteinander verbunden. Auf der eingesetzten Platine befindet sich u.a. ein Sensor vom Typ TS-NTC-103, der mit einem Widerstand einen Spannungsteiler bildet. Die temperaturabhängige Spannung wird dort abgegriffen und liegt als analoger Wert am ADC-Eingang eines IC Picaxe 08M2, der diesen Wert als digitalen Wert (0 ... 255) an eine RS232-Schnittstelle ausgibt. Das Programm RS232M greift auf die serielle Schnittstelle zu und liest u.a. den Datenstrom von Pin 2 (RxD).

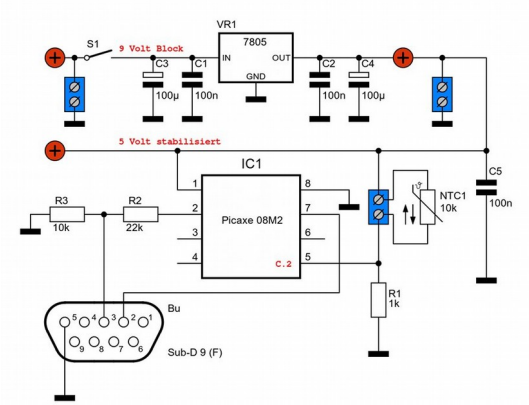

Abbildung 24.1.5.3.1: Schaltplan

Die Schaltung wurde von Stephan Mischnick (www.strippenstrolch.de) für das Gambas-Buch entworfen. Anschließend wurden die Platinen von ihm in einer Kleinserie gebaut und getestet.

24.1.5.3.1 Hinweise

Bei allen Projekten im Zusammenhang mit der Klasse SerialPort (gb.net) hat sich das folgende Vorgehen bewährt:

- Ermitteln, ob der Computer über serielle Schnittstellen verfügt. Ist das der Fall, so wird eine Liste der seriellen Schnittstellen erzeugt, in der die Pfad-Namen stehen wie zum Beispiel /dev/ttyUSB0 bei einem USB-RS232-Adapter oder /dev/ttyS0 bei einer echten seriellen Schnittstelle.
- Ermitteln, ob der Benutzer das Recht besitzt, auf serielle Schnittstellen zugreifen zu dürfen.
- Die Initialisierung der ausgewählten seriellen Schnittstelle umfasst drei Schritte: (1) Port-Namen festlegen, (2) Übertragungsparameter festlegen und (3) Datenfluss-Kontrolle festlegen.
- Seriellen Port öffnen: Nach dem erfolgreichen Öffnen geht SerialPort.Status auf Net.Active.
- Daten auslesen: Das Ereignis SerialPort_Read() wird ausgelöst, wenn Daten vom seriellen Port gelesen werden können. Option 1: Daten temporär in einer Variablen speichern, Option 2: Daten in einem geeigneten Format anzeigen und Option 3: Daten permanent speichern (Datei).
- Daten senden – wenn das erforderlich ist.
- Seriellen Port schließen.

Im vorgestellten Projekt wird das skizzierte Vorgehen konsequent umgesetzt. Auf die Option 3 wird verzichtet, da diese Option in einem weiteren Projekt eingesetzt wird. Daten werden nicht gesendet.

Der Quelltext von FMain.class wird vollständig angegeben und anschließend kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Private $bAllowedClose As Boolean
[4]  Private $aAdapterList As String[]
[5]  Private $sTemperatureDigit As String
[6]  Public TTimer As Timer
[7]
[8]  Public Sub Form_Open()
[9]
[10]   Dim sLine As String
[11]
[12]   FMain.Center()
[13]   FMain.Resizable = False
```

```
[14]   SetLEDColor("gray")
[15]   btnStart.Enabled = False
[16]
[17]   If MRS232.CheckPermission() = False Then
[18]      $bAllowedClose = True
[19]      FMain.Close()
[20]   Endif
[21]
[22]   $aAdapterList = MRS232.GetRS232Devices()
[23]
[24]   If $aAdapterList.Count = 0 Then
[25]      cmbRS232PortName.Add(("No RS232 device detected!"))
[26]      SetLEDColor("red")
[27]      cmbRS232PortName.Enabled = False
[28]      lblRS232Parameters.Visible = False
[29]      $bAllowedClose = True
[30]      MRS232.SendSound("error.ogg")
[31]   Else
[32]      For Each sLine In $aAdapterList
[33]        cmbRS232PortName.Add(sLine)
[34]      Next
[35]      btnStart.Enabled = True
[36]      lblRS232Parameters.Visible = True
[37]      lblRS232Parameters.Enabled = False
[38]      SetLEDColor("red")
[39]   Endif
[40]
[41] End
[42]
[43] Public Sub btnStart_Click()
[44]
[45] ' SetRS232PortName
[46]   If hRS232.Status = Net.Active Then hRS232.Close()
[47]   hRS232.PortName = cmbRS232PortName.Text
[48] ' SetRS232TransmissionParameters
[49]   hRS232.Speed = 4800
[50]   hRS232.DataBits = SerialPort.Bits8
[51]   hRS232.Parity = SerialPort.None
[52]   hRS232.StopBits = SerialPort.Bits1
[53] ' SetDataFlowControl
[54]   hRS232.FlowControl = SerialPort.None
[55]
[56]   MRS232.ShowRS232Parameters(hRS232, lblRS232Parameters)
[57]
[58]   OpenRS232Port()
[59]
[60]   cmbRS232PortName.Enabled = False
[61]   btnStart.Enabled = False
[62]
[63]   TTimer = New Timer As "TTimer"
[64]   TTimer.Delay = 500 * 1 ' The temperature is read out every 500ms
[65]   TTimer.Start()
[66]
[67]   MRS232.SendSound("on.wav")
[68]
[69] End
[70]
[71] Public Sub hRS232_Read()
[72]
[73]   Read #hRS232, $sTemperatureDigit, Lof(hRS232)
[74]
[75] End
[76]
[77] Public Sub TTimer_Timer()
[78]
[79]   ShowData($sTemperatureDigit)
[80]
[81] End
[82]
[83] Private Sub ShowData(sTemperature As String)
[84]
[85]   If $sTemperatureDigit <> "" Then
[86]      lblShowTemperature.Text = Asc(sTemperature) & " °C"
[87]      SetLEDColor("green")
[88]   Else
[89]      lblShowTemperature.Text = "--- °C"
[90]      SetLEDColor("red")
[91]   Endif
[92]
[93]   $sTemperatureDigit = ""
[94]
[95] End
[96]
[97] Private Sub OpenRS232Port()
```

```
[98]
[99]   Repeat
[100]      Try hRS232.Open(1)
[101]      If Error Then
[102]         Message.Error(("Error when opening the RS232 port!"))
[103]         Return
[104]      Endif
[105]    Until hRS232.Status = Net.Active
[106]
[107]  End
[108]
[109]  Private Sub SetLEDColor(sLEDColor As String)
[110]
[111]    pboxStatus.Picture = Picture["LED/led_" & sLEDColor & ".svg"]
[112]
[113]  End
[114]
[115]  Public Sub Form_Close()
[116]
[117]    If $bAllowedClose = False Then
[118]       If Message.Question(("Finish this experiment?"), ("Yes"), ("No")) = 1 Then
[119]          If hRS232 And If hRS232.Status = Net.Active Then hRS232.Close()
[120]       Else
[121]          Stop Event
[122]       Endif
[123]    Endif
[124]
[125]  End
```

Kommentar

- In den Zeilen 17 bis 20 wird ermittelt, ob der aktuelle Benutzer Zugriff auf die seriellen Schnittstellen hat. Die Funtion *CheckPermission()* befindet sich wie die Funktion GetRS232Devices() in einem Modul MRS232.module. Dessen Quelltext finden Sie im Projektordner des eingesetzten Projektes. Das Projekt-Archiv finden Sie im Download-Bereich.
- Die Auswertung des Funktionswertes der Funktion GetRS232Devices() erfolgt in den Zeilen 24 bis 39.
- Wenn der Benutzer zwar Zugriff auf serielle Schnittstellen hat, aber der Computer keine seriellen Schnittstellen besitzt, dann wird das angezeigt sowie optisch und akustisch signalisiert. Sie können das Programmfenster dann schließen:

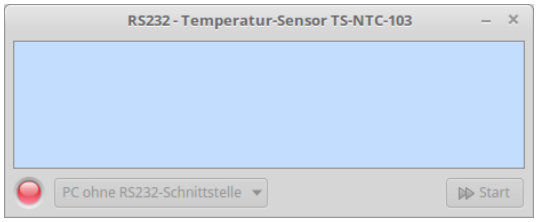

Abbildung 24.1.5.3.2: Fehlstart ...

- In den Zeilen 45 bis 54 werden der Portname, die Übertragungsparameter und die Art der Datenfluss-Steuerung festgelegt.
- Wenn der Benutzer Zugriff auf serielle Schnittstellen hat und der Computer über mindestens eine serielle Schnittstelle verfügt, dann können Sie die Messung starten:

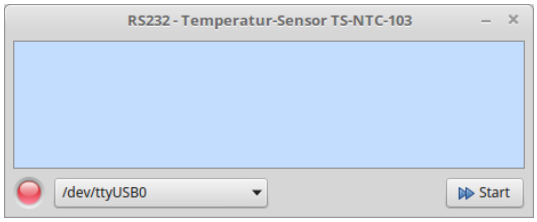

Abbildung 24.1.5.3.3: Startklar

- Nach dem Start wird der serielle Port (Zeilen 97 bis 107) geöffnet. Liefert der ADC Temperaturwerte, dann werden diese ausgelesen und in der globalen Variablen $sTemperatureDigit gespeichert.
- Die formatierte Anzeige wird über die Prozedur ShowData(…) in den Zeilen 83 bis 95 realisiert:

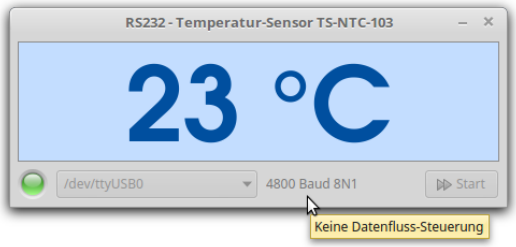

Abbildung 24.1.5.3.4: Anzeige der Temperatur und Einheit

- In der Zeilen 56 werden die eingestellte Baudrate, die gesetzten Übertragungsparameter und die Art der verwendeten Datenfluss-Steuerung erfasst und in einem Label in einem üblichen Format transparent angezeigt. Die Art der Datenfluss-Steuerung wird über einen Tool-Tipp des Labels sichtbar. Notwendig sind diese Anzeigen nicht.

Die Anzeige geht auf "- - - °C" , wenn

- die Stromversorgung abgeschaltet wird oder
- die Verbindung der seriellen Schnittstellen von Computer und Platine getrennt wird oder
- wenn während der Messung der USB-RS232-Adapter vom Computer abgezogen wird. Die Anzeige geht erst nach einigen Sekunden auf "- - - °C", wenn RTS vermutlich durch den Treiber auf "Not Ready To Send" gesetzt wurde.

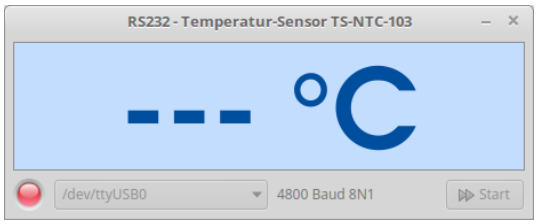

Abbildung 24.1.5.3.5: Es werden keine Daten ausgelesen