21.3.4 Projekt – 'Ping'

Das Projekt 'ping_p' ergänzt die Ausführungen im Kapitel 21.3.1 zum Beispiel 3. Während im genannten Beispiel-Projekt die *Quick-Syntax* der Instruktionen SHELL und EXEC eingesetzt wurde, wird jetzt der gestartete Prozess beobachtet und Daten sowohl aus der Standard-Ausgabe und als auch aus der Standard-Fehlerausgabe des Prozesses gelesen und angezeigt. Daten werden nicht an den Prozess übergeben, weil es sich bei dem Programm 'ping' *nicht* um ein *interaktives* Programm handelt. Der *aktive* Prozess kann beendet werden. Das wird im Projekt auch gezeigt.

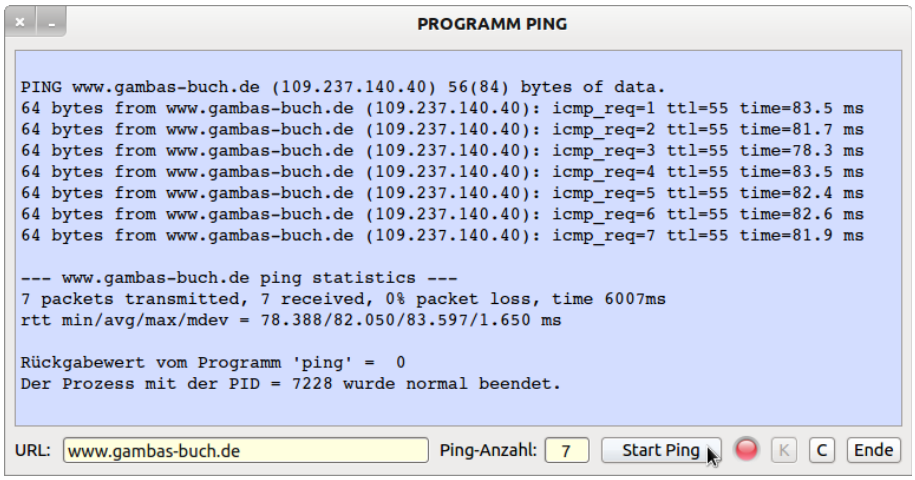

Abbildung 21.3.4.1: GUI-Entwurf für das Programm 'ping' – Programm beendet

Im Bild 21.3.4.1 sehen Sie die vollständigen Daten aus der Standard-Ausgabe des Prozesses nach dem regulären Ende des externen Konsolenprogramms 'ping' nach genau 7 Pings und eine eingefügte Benachrichtigung. Solange der gestartete Prozess aktiv ist, werden die eintreffenden Antwort-Daten asynchron nacheinander ausgegeben:

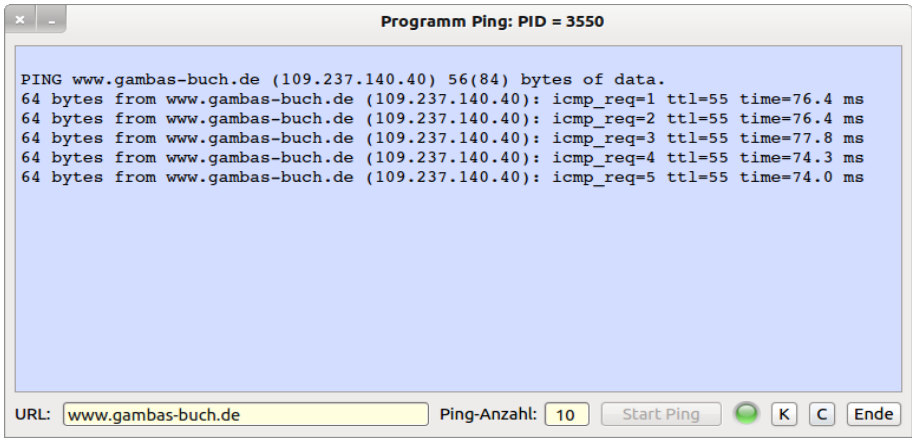

Abbildung 21.3.4.2: GUI-Entwurf für das Programm 'ping' – Programm zur Laufzeit

Prozess-Fehler-Ausgaben – in rot markiert – werden bei dem vorgestellten Projekt mit den minimal gesetzten Parametern bereits beim Start des Prozesses erkannt und in einer Fehlermeldung formatiert ausgegeben:

URL: www.gambas-buch.de
Count: -4

```
PROZESS-FEHLER!
ping: bad number of packets to transmit.

Rückgabewert vom Programm 'ping' =  2
Der Prozess mit der PID = 3423 wurde normal beendet.
```

URL: www.gambas+buch.de
Count: 4

```
PROZESS-FEHLER!
ping: unknown host www.gambas+buch.de

Rückgabewert vom Programm 'ping' =  2
Der Prozess mit der PID = 3970 wurde normal beendet.
```

Wenn man den gestarteten Prozess mit der Methode *$hPing.Kill()* beendet – ausgelöst durch einen Klick auf den Button mit der Beschriftung K – dann wird das Ereignis *Kill* erkannt und im Event-Handler *myPingProcess_Kill()* behandelt, formatiert und angezeigt:

```
Rückgabewert vom Programm 'ping' =  9
Der Prozess mit der PID = 3984 wurde beendet! (SIGKILL)
```

Quelltext der relevanten Prozeduren:

```
Public Sub btnProcessKill_Click()

  If $hPing Then $hPing.Kill()
  ' Alternative über System-Aufruf:
  ' If $hPing Then Shell "kill -s 9 " & $hPing.Id
  ' Endif ' $hPing existiert ?
  txaOutput.Foreground = Color.Red
  txaOutput.Clear

End ' btnProcessKill_Click()

Public Sub myPingProcess_Kill()
  txaOutput.Insert(gb.NewLine)
  txaOutput.Insert("Rückgabewert '" & sProgrammName & "' =  " & $hPing.Value & gb.NewLine)

  Select Case $hPing.State
  Case 0
    txaOutput.Insert("Prozess (PID = " & $hPing.Id & " ) wurde normal beendet." & gb.NewLine)
  Case 1
    txaOutput.Insert("Prozess (PID = " & $hPing.Id & " ) wurde gestoppt!" & gb.NewLine)
  Case 2
    txaOutput.Insert("Prozess (PID = " & $hPing.Id & " ) beendet! (SIGKILL)" & gb.NewLine)
  End Select ' $hProcess.State

  SetLEDColor("red")
  FMain.Text = "PROGRAMM PING"
  btnProcessKill.Enabled = False
  btnPingProcessStart.Enabled = True
End ' hPing_Kill()
```

Den vollständigen Quelltext finden Sie im Download-Bereich des Gambas-Projekts.