arduino Code Testing Qu

Automatisation qualvoll

engelbert gruber

19.04.2020
1ST GRADE SCIENCE

Vorwort (nachher geschrieben)

Dieser Bericht ist lang und mühsam.

Weil Programme die nicht zum Testen mit Unittests geschrieben worden sind, schwierig zu testen sind. (Es gibt auch hier die Ausnahmen und Gründe.)

Aber man sieht, dann doch dass die Probleme die man hat in dem unnötigem Code bestehen, Cut'n'paste.

Es gibt viele Code smells, kümmern muss man sich um doppelten Code und doppelte Daten. Programmieren ist strukturieren in Code **und** Daten. Deshalb die vielen "Datenstrukturen und Algorithmen" Bücher.

21:54 continued from pre

INTRODUCTION

Die Vorarbeiten sind im vorigen -"pre"-Dokument beschrieben.

Dort waren wir beim Erkennen des Programmablaufs über den Seiteneffekt, dass PinModes ändern. Das Programm setzt:

```
pinMode(s1, INPUT);
pinMode(s2, INPUT);
```

S1 und s2 sind aber nur die Nummern der Pins nicht die Werte. Das müssen wir erst einbauen. Arduino-simu:

```
int simu_pinMode[16];
void pinMode(int pin, int mode) {
   simu_pinMode[pin] = mode;
}
```

Und im test.cpp:

```
simu_pinMode[1] = 99;  // invalid
simu_analogIn = 100;
loop();
if (simu_pinMode[1] == 0) {
  std::cout << "OK" << std::endl;
  return 0;
}
std::cout << "ERROR" << std::endl;
```

runtest:

```
100
MB = 5V | U = 4.99512V
1023OK
```

Zweiter Test

Wir müssen im test.cpp mehrere Test ausführen, etwa so:

Test.cpp

```
// Testrahmen für ein arduino Programm

#include "arduino-simu.h"
#include "arduino-voltmeter/arduino-voltmeter.ino"

int main() {
   int tests = 0;
   int errors = 0;
   // test 1
   Tests++;
   simu_pinMode[1] = 99;  // invalid
```

```
    simu_analogIn = 100;
    loop();
    if (simu_pinMode[1] != 0) {
      Errors++;
    }
    // summary
    std::cout << std::endl << "-------------" << std::endl;
    if (errors > 0) {
      std::cout << "ERROR. " << errors << " of " << tests
                << " tests failed." << std::endl;
      return 1;
    }
    std::cout << "OK. " << tests << " passed." << std::endl;
    return 0;
}
```

Etwas davon könnten wir uns durch Verwenden eines unittest-Frameworks ersparen, aber das geht jetzt noch so.

Wir bekommen mit cut'n'paste :

```
// test 1: Messbereich 5V
tests++;
simu_pinMode[1] = 99;  // invalid
simu_pinMode[2] = 99;  // invalid
simu_analogIn = 100;
simu_cnt = 0;
loop();
if ((simu_pinMode[1] != 0)||(simu_pinMode[2] != 0)) {
  errors++;
}
// test 2: Messbereich 10V
tests++;
simu_pinMode[1] = 99;  // invalid
simu_pinMode[2] = 99;  // invalid
simu_analogIn = 300;
simu_cnt = 0;
loop();
```

```
if ((simu_pinMode[1] != 1)||(simu_pinMode[2] != 0)) {
  errors++;
}
// test 3: Messbereich 20V
tests++;
simu_pinMode[1] = 99;  // invalid
simu_pinMode[2] = 99;  // invalid
simu_analogIn = 500;
simu_cnt = 0;
loop();
if ((simu_pinMode[1] != 0)||(simu_pinMode[2] != 1)) {
  errors++;
}
// test 4: Messbereich +20V
tests++;
simu_pinMode[1] = 99;  // invalid
simu_pinMode[2] = 99;  // invalid
simu_analogIn = 900;
simu_cnt = 0;
loop();
if ((simu_pinMode[1] != 1)||(simu_pinMode[2] != 1)) {
  errors++;
}
```

Runtest meldet:

```
  -------------
  OK. 4 passed.
```

(svn ci)

Remove duplicates

Beinahe alles ist da eine Wiederholung. Das stinkt schon wieder, ab in eine Schleife.

Wir richten uns zuerst die Daten in ein Array und klappern das mit einer for-Schleife ab.

```
int test_set[][3] = {
  // analogIn, pinMode1 und 2 nachher
  {100, 0, 0}, // test 1: Messbereich 5V
```

```
      {300, 1, 0}, // test 2: Messbereich 10V
      {500, 0, 1}, // test 3: Messbereich 20V
      {900, 1, 1}, // test 4: Messbereich +20V
      };
   for (int i=0; i<(sizeof(test_set)/sizeof(test_set[0])); i++) {
      tests++;
      simu_pinMode[1] = 99;  // invalid
      simu_pinMode[2] = 99;  // invalid
      simu_analogIn = test_set[i][0];
      simu_cnt = 0;
      loop();
      if ((simu_pinMode[1] != test_set[i][1])||(simu_pinMode[2] !=
test_set[i][2])) {
        errors++;
        std::cout << "ERROR" << std::endl;
      }
   }
```

Das macht das selbe aber Änderungen, wie die Asugabe von ERROR wenn ein Test schiefgeht muss nur einmal gemacht werden.

(svn ci)

Seltsames

Runtest gibt folgendes aus:

```
      100
      MB = 5V | U = 4.99512V
      1023300
      MB = 10V |
      U = 9.99023V
      500
      MB = 20V | U = 19.9805V
      1023900
      !!!MB > 20V!!!

      -------------
```

```
OK. 4 passed.
```

Die Ausgabe "`MB = 10V | U = 9.99023V`" geht über zwei Zeilen und danach steht nicht 1023 ?

Beim Messbereich 5V steht in der while-Schleife:

```
Serial.print("MB = 5V | ");
in = analogRead(inputpin);
u = in * 5 / 1024;
Serial.print("U = ");
Serial.print(u);
Serial.println("V");
Serial.print(in);
delay(1000); //damit nicht alles "durchrauscht"
```

Und der 10V Messbereich:

```
Serial.println("MB = 10V | ");
in = analogRead(inputpin);
u = in * 10 / 1024;
Serial.print("U = ");
Serial.print(u);
Serial.println("V");
delay(1000);//damit nicht alles "durchrauscht"
```

Der MB wird mit println ausgegeben und nach println("V") fehlt print(in).

Remove duplicates

Der Inhalt der Schleife ist cut'n'paste. Wir extrahieren eine Funktion

```
void measure(int max_voltage) {
  Serial.print("MB = ");
  Serial.print(max_voltage);
  Serial.print("V | ");
  in = analogRead(inputpin);
  u = in * max_voltage / 1024;
  Serial.print("U = ");
  Serial.print(u);
  Serial.println("V");
```

```
}
```

Im 5V Bereich:

```
while (analogRead(inputpin) < 1023)
{
   measure(5);
   delay(1000); //damit nicht alles "durchrauscht"
}
```

Runtest liefert immer noch:

```
100
MB = 5V | U = 4.99512V
300
MB = 10V |
U = 9.99023V
500
MB = 20V | U = 19.9805V
1023900
!!!MB > 20V!!!

-------------
OK. 4 passed.
```

Wenn wir in allen drei Schleifen die Funktion measure verwenden

```
100
MB = 5V | U = 4.99512V
300
MB = 10V | U = 9.99023V
500
MB = 20V | U = 19.9805V
900
!!!MB > 20V!!!

-------------
OK. 4 passed.
```

Immer noch wie vorher. Svn ci.

Wir haben jetzt:

```
  else if (in <= 204)  //MB = 5V
  {
     pinMode(s1, INPUT);
     pinMode(s2, INPUT);
     while (analogRead(inputpin) < 1023)
     {
        measure(5);
        delay(1000);//damit nicht alles "durchrauscht"
     }
  }
  else if (in > 204 && in <= 409)  //MB = 10V
  {
     pinMode(s1, OUTPUT);
     digitalWrite(s1, 0);
     pinMode(s2, INPUT);
     while (analogRead(inputpin) < 1023 && analogRead(inputpin)>=
526)
     {
        measure(10);
        delay(1000);//damit nicht alles "durchrauscht"
     }
  }
  else if (in > 409 && in <= 818)  //MB = 20V
  {
     pinMode(s1, INPUT);
     pinMode(s2, OUTPUT);
     digitalWrite(s2, 0);
     while (analogRead(inputpin) < 1023 && analogRead(inputpin)>=
526)
     {
        measure(20);
        delay(1000);//damit nicht alles "durchrauscht"
     }
```

```
}
```

Wenn wir die Umschaltwerte und pinModes in ein Array stecken wie bei den Tests könnte das noch kleiner werden.

Noch komischer

Vorher muss ich aber das mit dem Wert noch verstehen.

Runtest gibt folgendes aus

```
100
MB = 5V | U = 4.99512V
300
MB = 10V | U = 9.99023V
500
MB = 20V | U = 19.9805V
900
!!!MB > 20V!!!
```

Der analogRead-Wert von 100 ist beinahe 5V, bei einem 10bit-Wandler sollte das aber 0,5V sein: 100/1024*5. Da ist etwas mit der Skalierung im Argen:

```
in = analogRead(inputpin);
u = in * max_voltage / 1024;
```

Die Simulation liefert den falschen Wert. Wenn erst der 4. 1023 ist dann kommt folgendes heraus:

```
100
MB = 5V | U = 0.488281V
300
500
900
!!!MB > 20V!!!
```

Besser, aber die anderen Bereiche funktionieren nicht mehr.

Weil manchmal ein analogRead mehr gemacht wird:

```
while (analogRead(inputpin) < 1023)
{
  measure(5);
```

```
            delay(1000); //damit nicht alles "durchrauscht"
          }
        }
        else if (in > 204 && in <= 409)  //MB = 10V
        {
            pinMode(s1, OUTPUT);
            digitalWrite(s1, 0);
            pinMode(s2, INPUT);
            while (analogRead(inputpin) < 1023 && analogRead(inputpin)>= 526)
```

Eigentlich brauchen wir jedesmal nur einen neuen Wert holen.

```
    float measure(int max_voltage) {
       float in;
       Serial.print("MB = ");
       Serial.print(max_voltage);
       Serial.print("V | ");
       in = analogRead(inputpin);
       u = in * max_voltage / 1024;
       Serial.print("U = ");
       Serial.print(u);
       Serial.println("V");
       return in;
    }
```

Und in der loop-Funktion:

```
   while (in < 1023)
   {
      in = measure(5);
      delay(1000); //damit nicht alles "durchrauscht"
   }
}
else if (in > 204 && in <= 409)  //MB = 10V
{
   pinMode(s1, OUTPUT);
   digitalWrite(s1, 0);
   pinMode(s2, INPUT);
   in = analogRead(inputpin);  // mit richtiger Skalierung
   while (in < 1023 && in >= 526)
```

```
  {
     in = measure(10);
```

Wenn wir das bei allen ändern und uns von analogRead die Werte in Klammern ausgeben lassen, gibt runtest folgendes aus:

```
(100)100
MB = 5V | (100)U = 0.488281V
MB = 5V | (100)U = 0.488281V
MB = 5V | (1023)U = 4.99512V
(300)300
(300)(500)500
(500)(900)900
```

Mit analogRead 300 kommen wir in den 10V Messbereich, aber auch sofort wieder heraus weil die Schleife :

```
while (in < 1023 && in >= 526)
```

Völlig korrekt davon ausgeht, dass der darunterliegende Messbereich besser geeignet ist.

Wir haben hier ein Problem mit dem Testerzeugen, nicht mit dem Programm (ausser, dass da vieles doppelt und dreifach ist). Wir können das Ganze umdrehen.

```
else if (in <= 204)  //MB = 5V
{
  pinMode(s1, INPUT);
  pinMode(s2, INPUT);
  do {
   in = measure(5);
   delay(1000); //damit nicht alles "durchrauscht"
  }
  while (in < 1023);
}
else if (in > 204 && in <= 409)  //MB = 10V
{
  pinMode(s1, OUTPUT);
  digitalWrite(s1, 0);
  pinMode(s2, INPUT);
  do {
```

```
      in = measure(10);
      delay(1000);//damit nicht alles "durchrauscht"
    }
    while (in < 1023 && in >= 526);
  }
  else if (in > 409 && in <= 818)  //MB = 20V
  {
    pinMode(s1, INPUT);
    pinMode(s2, OUTPUT);
    digitalWrite(s2, 0);
    do {
      in = measure(20);
      delay(1000);//damit nicht alles "durchrauscht"
    }
    while (in < 1023 && in >= 526);
  }
}
```

Dann bekommen wir von runtest:

```
      (100)100
      MB = 5V | (100)U = 0.488281V
      MB = 5V | (100)U = 0.488281V
      MB = 5V | (1023)U = 4.99512V
      (300)300
      MB = 10V | (300)U = 2.92969V
      (500)500
      MB = 20V | (500)U = 9.76562V
      (900)900
```

Das schaut gut aus. Wir stellen analogRead wieder zurück, dreimal den vorgegebenen wert, dann 1023. Und den Debugoutput (100) … nicht mehr.

Remove duplicates

```
  else if (in <= 204)  //MB = 5V
  {
    pinMode(s1, INPUT);
```

```
    pinMode(s2, INPUT);
    do {
     in = measure(5);
     delay(1000); //damit nicht alles "durchrauscht"
    }
    while (in < 1023);
  }
  else if (in > 204 && in <= 409)  //MB = 10V
  {
    pinMode(s1, OUTPUT);
    digitalWrite(s1, 0);
    pinMode(s2, INPUT);
    do {
     in = measure(10);
     delay(1000);//damit nicht alles "durchrauscht"
    }
    while (in < 1023 && in >= 526);
  }
  else if (in > 409 && in <= 818)  //MB = 20V
  {
    pinMode(s1, INPUT);
    pinMode(s2, OUTPUT);
    digitalWrite(s2, 0);
    do {
     in = measure(20);
     delay(1000);//damit nicht alles "durchrauscht"
    }
    while (in < 1023 && in >= 526);
  }
```

Nach dem Schalten der pinModes sind die while Schleifen sehr ähnlich.

Es gibt einen max-Voltage-Wert und im 5V-Messbereich gibt es keinen Minimalwert.

```
int max_voltage = 1;
int min_in = 0;
```

Und die Schleifen

```
  else if (in <= 204)  //MB = 5V
  {
    pinMode(s1, INPUT);
    pinMode(s2, INPUT);
    max_voltage = 5;
    min_in = 0;
  }
  else if (in > 204 && in <= 409)  //MB = 10V
  {
    pinMode(s1, OUTPUT);
    digitalWrite(s1, 0);
    pinMode(s2, INPUT);
    max_voltage = 10;
    min_in = 526;
  }
  else if (in > 409 && in <= 818)  //MB = 20V
  {
    pinMode(s1, INPUT);
    pinMode(s2, OUTPUT);
    digitalWrite(s2, 0);
    max_voltage = 20;
    min_in = 526;
  }
  do {
      in = measure(max_voltage);
      delay(1000);//damit nicht alles "durchrauscht"
  }
  while (in < 1023 && in >= min_in);
```

Runtest:

```
    100
    MB = 5V | U = 0.488281V
    MB = 5V | U = 4.99512V
    300
    MB = 10V | U = 2.92969V
    500
```

```
MB = 20V | U = 9.76562V
900
!!!MB > 20V!!!
MB = 1V | U = 0.878906V
MB = 1V | U = 0.999023V

-------------
OK. 4 passed.
```

Next maybe ToDo

- Kein measure wenn wir ausserhalb des Bereiches sind oder ausrechnen welche Spannung wir dort haben.
- Den analogRead-Wert nicht ausgeben.
- Der von analogRead gelesene Wert kann auch später in float umgewandelt werden.

0:25

CONCLUSION

Arbeit ist Energie und kommt nicht von Nichts.

REFERENCES

1. https://en.wikipedia.org/wiki/Duplicate_code