Machbarkeitsstudie PUK Erkennung

Versuchs Beschreibung:

Ein Puk wird etwa ein Zentimeter vor den Punkt auf das Band gelegt, wo der Messpunkt des Höhensensors ist. Es werden 400 Messwerte mit dem Programm „Height\_Sensor\_Test“ (unter ESEP2016 project: src/test zu finden) genommen. Die 400 Messwerte werden auf der Konsole ausgegeben und in dem Matlab-script „esep\_puks“ als Vektor mit passenden Variablennamen eingefügt. In dem Matlab-script werden einige Messungen zur Übersicht als Graph dargestellt.  
  
Es wird ein Algorithmus in der Matlab-funktion „puk\_algo“ entwickelt der anhand der Messwerte die Puks zuverlässig erkennen kann.  
Durch die Veränderung der Messwerte durch das Öffnen der Weiche wird eine Testreihe durchgeführt, in der die Weiche nach 100 genommen Messwerten geöffnet wird. Die Veränderung der Messwerte wird untersucht.

Puks im Test:

Sunshine: hoher Puk mit gelben Isotape umwickelt und Rillen in der Folge von außen nach innen: breite Rille, schmale Rille.

Innocence: hoher Puk mit weißen Isotape umwickelt und Rillen in der Folge von außen nach innen: breite Rille, breite Rille.

Deep\_Blue: hoher Puk mit blauen Isotape umwickelt und Rillen in der Folge von außen nach innen: schmale Rille, breite Rille, schmale Rille.

Black\_Beauty: hoher Puk mit schwarzen Isotape umwickelt und zwei schmalen Rillen.

Iron\_Core: hoher Puk mit Bohrung und Eisenkern (mit Rille).

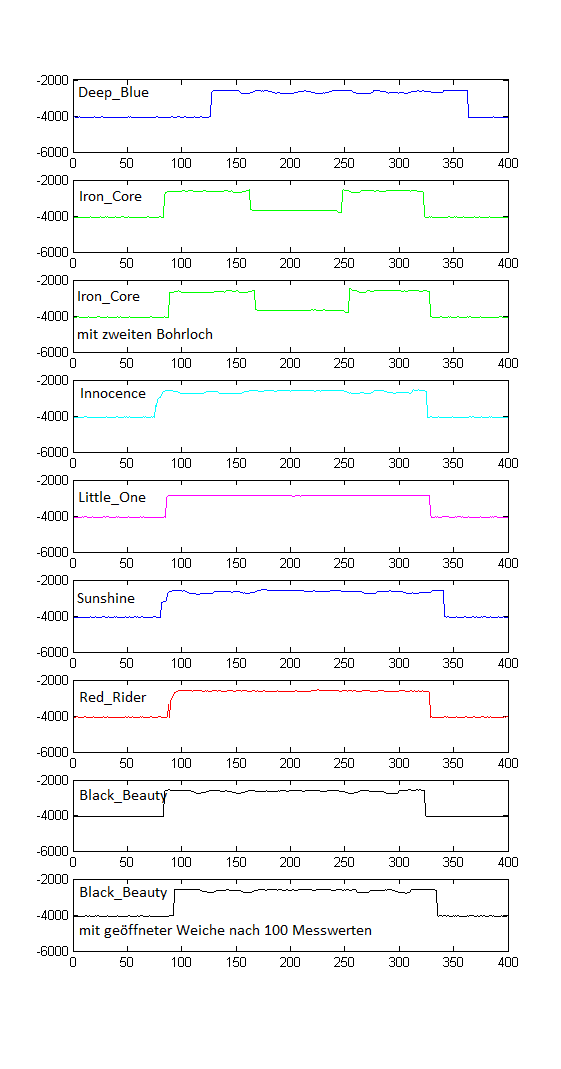
Red\_Rider: hoher, roter Puk mit Bohrung.

Little\_One: niedriger Puk.

Messwerte:

Die Messwerte sind dem Matlab-script „esep\_puks“ zu entnehmen.

Graph



Puk Kategorisierung Algorithmus & Bewertung der Messwerte

Die Idee ist die Werte gegen Grenzwerte zu prüfen und auf diese Weise Sprünge zwischen hoch (Normale Puk Höhe) und niedrig zu bestimmen.

Um wildes Springen durch Rauschen zu verhindern sind die Grenzwerte mit einem Abstand gewählt.

Durch Tests und Analysieren der Messwerte haben sich folgende Grenzwerte ergeben:

Obere Grenze = -2625 (Werte aus den mit -1 multiplizierten Vektoren)

Untere Grenze = -2695

Dadurch entstehen folgende Bit Code für die Puks:

deep\_blue = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0]

iron\_core = [1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Little\_One = [0, 0, 0, 0, 0, 0, 0 , 0, 0, 0, 0, 0, 0, 0, 0, 0]

red\_rider = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Innocence, Sunshine, Black\_Beauty = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0]

Auf diese Weise werden 5 Kategorien unterschieden. Um Innocence, Sunshine und Black\_Beauty zu unterscheiden muss die breite der Rillen bestimmt werden. Dazu wird der Index für die ersten Vier Sprünge auf 0 (niedrig) gespeichert. Von diesen Werten wird der Abstand bestimmt bis ein weiterer Grenzwert überschritten ist. Dieser Grenzwert wurde die bisher bestimmt und liegt bei -2660.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Puk Name | Rille | Begin | Wert | Ende | Wert | Abstand |
| Black\_Beauty | 1 (schmal) | 112 | -2701 | 124 | -2642 | 12 |
| Black\_Beauty | 4 (schmal) | 287 | -2728 | 299 | -2655 | 12 |
| Black\_Beauty | 2 (schmal) | 147 | -2694 | 259 | -2654 | 12 |
| Black\_Beauty | 3 (schmal) | 251 | -2699 | 264 | -2617 | 13 |
| Innocence | 1 (breit) | 98 | -2704 | 123 | -2643 | 25 |
| Innocence | 4 (breit) | 290 | -2702 | 312 | -2562 | 22 |
| Innocence | 2 (breit) | 130 | -2697 | 159 | -2641 | 25 |
| Innocence | 3 (breit) | 252 | -2705 | 276 | -2612 | 24 |

Begin, Ende und Abstand in Messwerten.

Der Tabelle ist zu entnehmen das in der Stichprobe der Abstand für schmale Rillen 12 bis 13 Messwerte beträgt und der Abstand der breiten Rillen 22 bis 25 Messwerte beträgt.

Um Sicherheit gegen Abweichungen zu schaffen werden die Grenzwerte weiter gewählt:

Schmale Rillen untere Grenze = 9 Messwerte

Schmale Rillen obere Grenze = 16 Messwerte

Breite Rillen untere Grenze = 20 Messwerte

Breite Rillen untere Grenze = 27 Messwerte

Durch Vergleichen der Rillenbreiten der Puks mit den Grenzwerten werden die Puks bestimmt. Um die Puks Innocence, Sunshine und Black\_Beauty nun unterscheiden zu können wird ihr Bit Code an den Indexen 15 und 16 (Matlab: Index startet mit 1) angepasst:

Innocence = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1]

Sunshine = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0]

Black\_Beauty = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1]