NWI-IMC065: Assignment 2

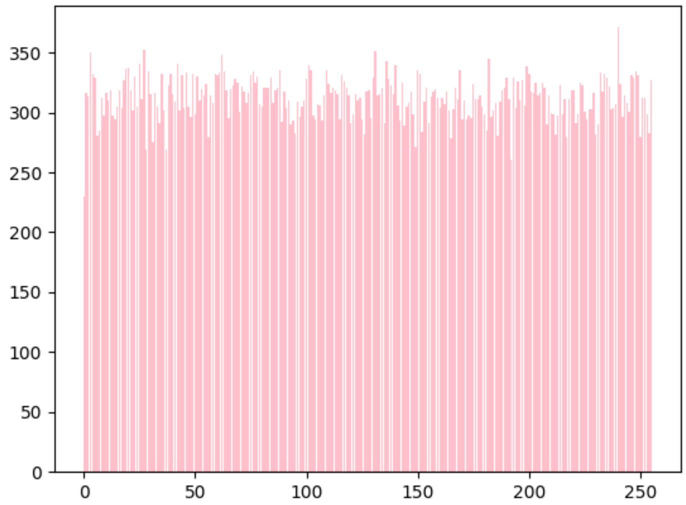
Wouter Doeland — s1034816
Wouter van Battum — s1011825

December 20, 2023

1 Testing Random Number Sequences

1.1 *Q1

- The plot for 20.000 values of `AES_OFB` is added below.



  The histogram is fairly uniformly spread, with a few outliers.

- This is AES in output feedback mode. It allows AES to be used as a stream cipher. In this case we're using the 'stream cipher' output as random numbers.

Q2

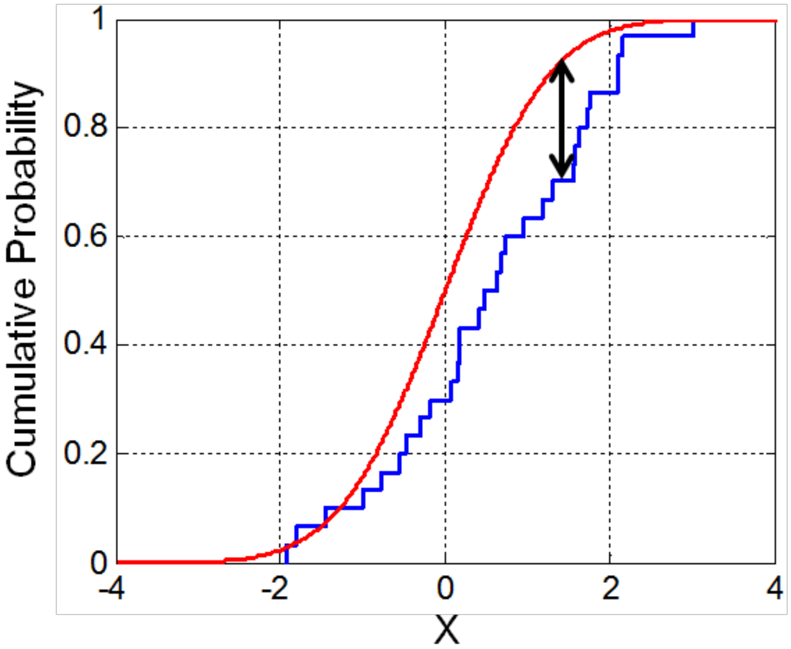
- The min-entropy per bit is 0.9688854802418052.
- The shannon-entropy per bit is 0.9997043710574444.

- The min-entropy is the most conservative way to measure unpredictability, while the shannon-entropy is the average way of measuring it. Because the worst case is so close to the average, there are not really many bad entries in the spread.

Q3

Small comment: if you want us to pick a specific test, put that BEFORE the exercise where we have to do the work, so we don't spend two hours figuring out/explaining the wrong algorithm.

- We choose test 4: $(6 + 5) \bmod 7 = 4$. So this is the "RGB Kolmogorov-Smirnov Test" (nr 204). The Kolmogorov-Smirnov Test compares an ideal distribution curve to the actual distribution curve. It then selects the variable with the maximum absolute distance between the ideal and actual curve. Dieharder doesn't use the K-S test directly but uses the Anderson-Darling and Kuiper statistical tests. These are slightly modified (the first adds a weight and the second takes a negative and positive absolute difference) versions of the K-S test.

  We take this illustration (by Bscan, CC0, `https://commons.wikimedia.org/w/index.php?curid=25222928`) to explain the test. In this case the red line is the ideal curve and the blue line is the actual curve. The test looks for the point with the biggest absolute distribution between the ideal and actual curve. In this case that is around $x = 1.5$, as illustrated by the black arrow.

-
  ```
  [wouter@fedora hw-sec-a2]$ dieharder -g 202 -d 204 -f Part\ 1/samples.txt
  #=============================================================================#
  #            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
  #=============================================================================#
     rng_name    |           filename             |rands/second|
  ```

```
     file_input|              Part 1/samples.txt|  1.01e+07  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
# The file file_input was rewound 1000 times
     rgb_kstest_test|   0|     10000|    1000|0.00000000|  FAILED
```

The test failed! It output a p-value of 0.000. We think it failed because the input file was too short and it had to be rewound many times. This breaks many tests.

Q4

- We picked tests 204 (K-S test) and 0 (birthday test). Below is the output of running both tests.

```
[wouter@fedora hw-sec-a2]$ dieharder -g 201 -d 204 -f Part\ 1/broken_sample.bin
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
 file_input_raw|        Part 1/broken_sample.bin|  9.49e+07  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
# The file file_input_raw was rewound 1000 times
     rgb_kstest_test|   0|     10000|    1000|0.00000000|  FAILED
[wouter@fedora hw-sec-a2]$ dieharder -g 201 -d 0 -f Part\ 1/broken_sample.bin
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
 file_input_raw|        Part 1/broken_sample.bin|  9.50e+07  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
# The file file_input_raw was rewound 692 times
   diehard_birthdays|   0|       100|     100|0.22962149|  PASSED
```

- Our generator generates a file by counting from 0 to 20000. Every number is hashed using sha256, of which we then take the first 4 bytes. This is the 32-bit number we take as input for Dieharder. These 32-bit numbers are put after each other in the binary file.
- The python script we use is added below:

```
import hashlib
```

```
OUTPUT_SAMPLES_FILE_NAME = 'Part 1/broken_sample.bin'
NUMBERS = 20000

output_samples = open(OUTPUT_SAMPLES_FILE_NAME, 'wb')

for count in range(0, NUMBERS):
    output_samples.write(hashlib.sha256(bytes(count)).digest()[:4])

output_samples.close()
```

Physically Unclonable Functions

Q1

For k we took 6 as $((5+6)\%7)+2=6$. We simulated the instances with the following command: `pufi = XORArbiterPUF(n=64, k=6, noisiness=0.1, seed=i)`, where `i` was in the set {1,2,. . . , 9,10} to make sure that each PUF instance was slightly different.

Q2

1. This can be found in `Part2.ipynb`.
2. The bias of the PUF instances are respectively: [0.5009, 0.5015, 0.49995, 0.50065, 0.4998, 0.50015, 0.5067, 0.4973, 0.49595, 0.4956] and the average bias is 0.49984999999999999. Uniformity is the distribution of 1's and 0's and if the PUF is uniform, the distribution is 50%, 50/50. If the PUF is biased, its distribution is not 50%, then it has more 0's than 1's or vice versa. Thus, the bias relates to the level of uniformity a PUF has, the closer the bias is to 0.5, the better the distribution of 1's and 0's. Therefore, the uniformity of the PUFs is good as it is almost 0.5.
3. The uniqueness of each PUF instance is respectively: [0.42684947, 0.44900737, 0.44962, 0.42149895, 0.41828421, 0.42629263, 0.42972211, 0.44838421, 0.44967421, 0.43861474], and the average uniqueness was 0.4357947894736842. The uniqueness histogram can be found in Figure 1. The uniqueness is okay but could be better. Because the ideal uniqueness is 0.5 and the average uniqueness of the PUFs could be a lot closer to 0.5.
4. The reliability histograms can be found in Figures 2 and 3.

Q3

When we increase the number of measurements, we can see that the Hamming distance between the temporary majority-voted response and the golden response from Q2 decreases. This is in line with the reliability of features of PUFs, meaning that the functionality should be repeatable at different instants of time under various environments. We can also see that the HD of the temporal majority-voted response from 15 measurements and the golden response from Q2 is 0. This is because they are the same, as the are computed from the same dataset. All the HDs can be found in Table 1.

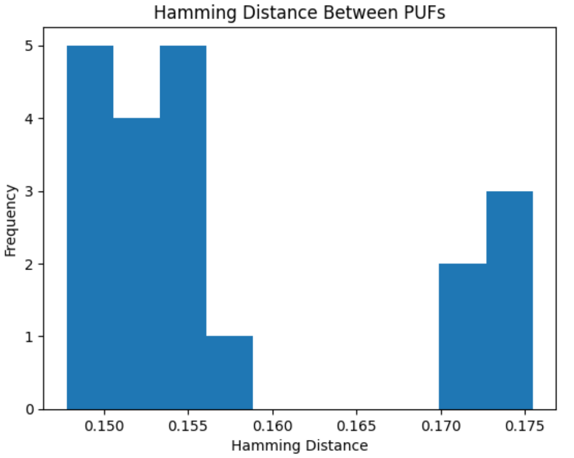


Figure 1: Histogram of the HD of the golden response obtained from each pair of instances

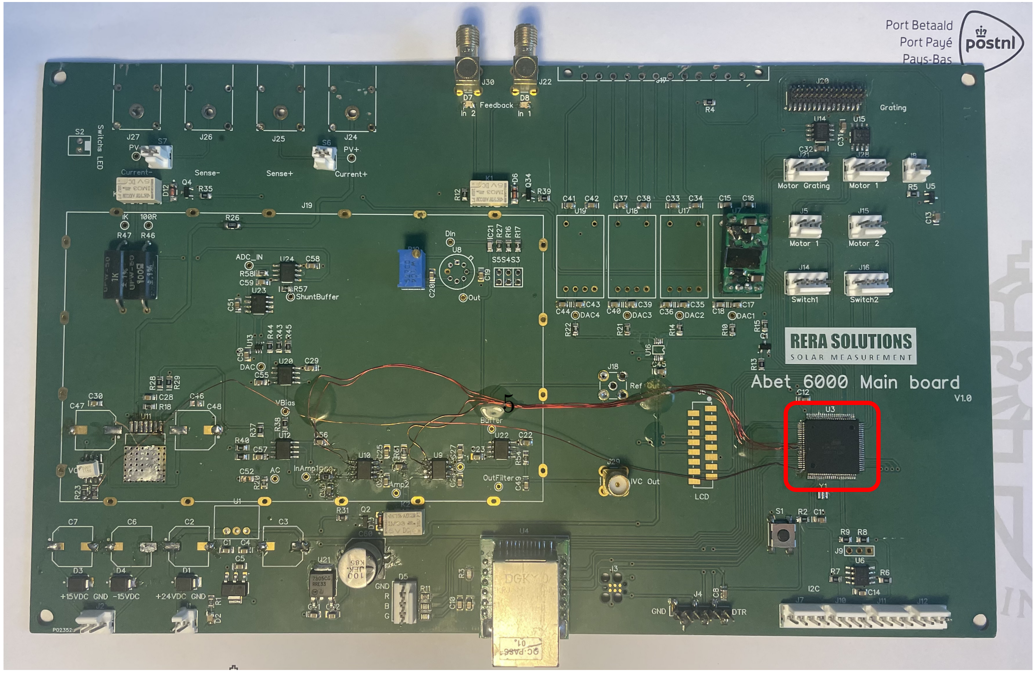
Instance	HD - 5	HD - 15	HD - 25
PUF 1	0.07525	0.0	0.038
PUF 2	0.082	0.0	0.04195
PUF 3	0.0835	0.0	0.0426
PUF 4	0.07425	0.0	0.0379
PUF 5	0.07185	0.0	0.03625
PUF 6	0.0768	0.0	0.0375
PUF 7	0.0769	0.0	0.03935
PUF 8	0.0818	0.0	0.04215
PUF 9	0.08315	0.0	0.0433
PUF 10	0.07935	0.0	0.04215

Table 1: The HD from the temporal majority-voted responses with the golden response

Mystery Board

Q1

- The box shows the microcontroller



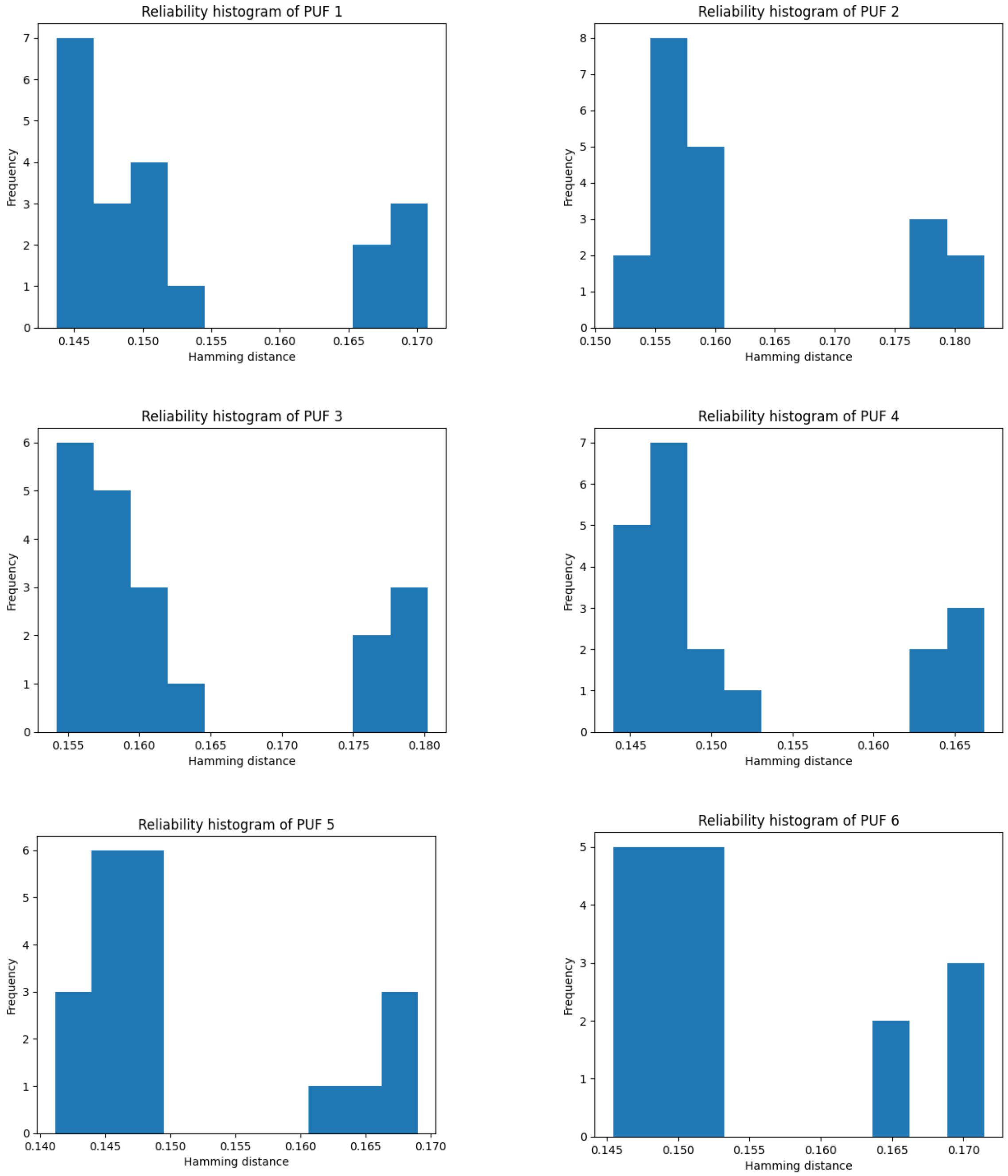


Figure 2: Reliability histograms part 1

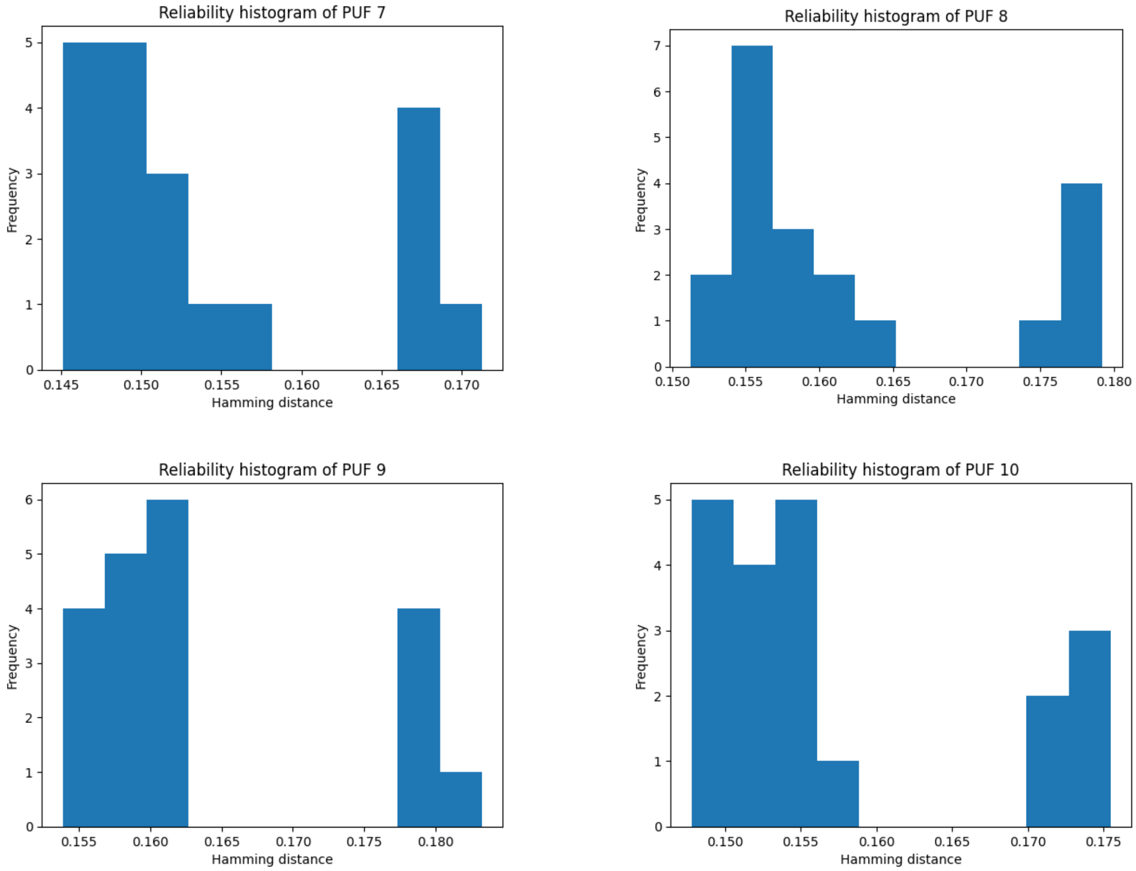
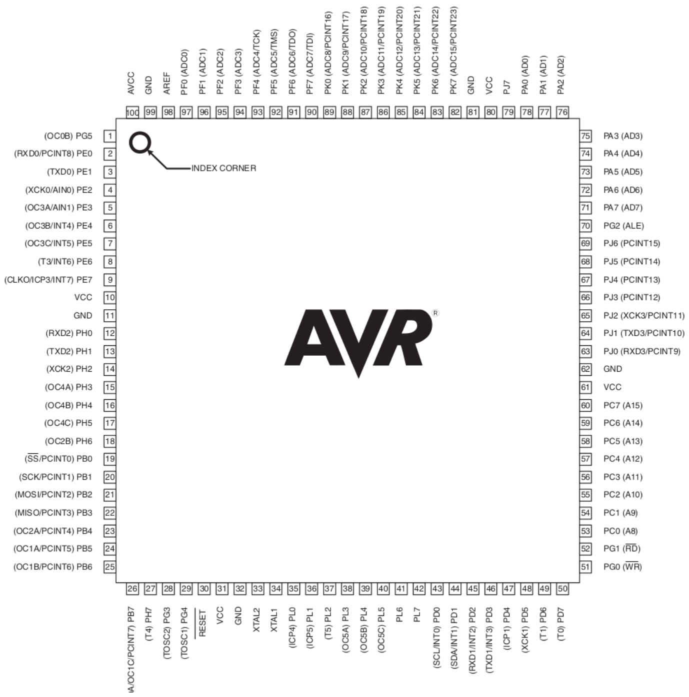


Figure 3: Reliability histograms part 2

- It is an ATMEGA2560 by Atmel
- The ground pins are on: 11, 32, 62, 81, 99. Below is a screenshot of the pinout from the datasheet.

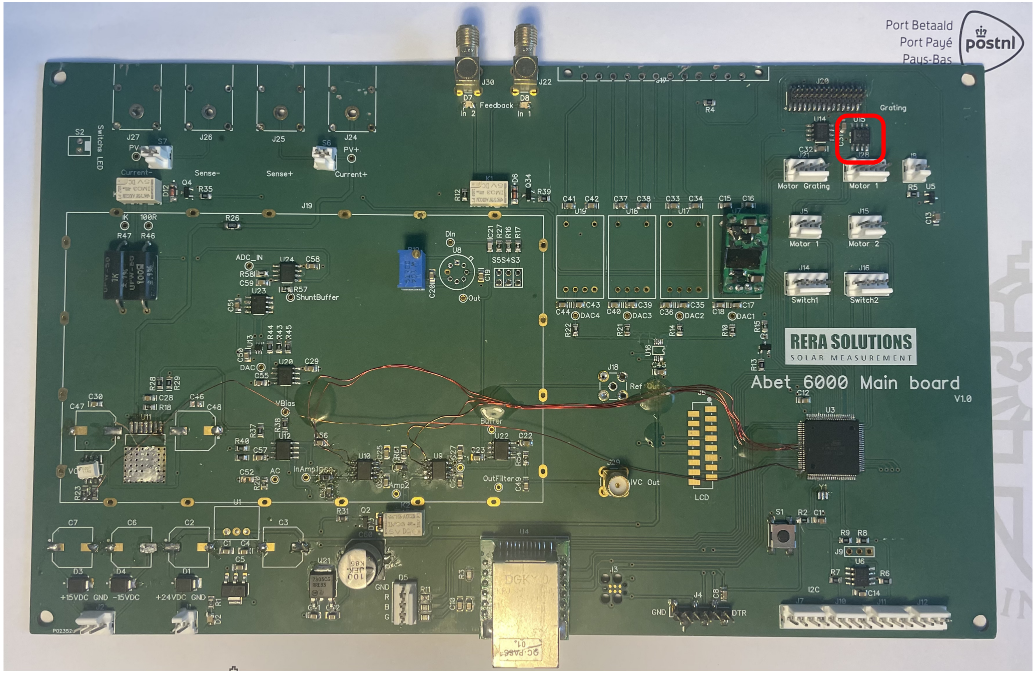
Figure 1-1. TQFP-pinout ATmega640/1280/2560



- The chip contains 8Kbytes SRAM according to the datasheet.

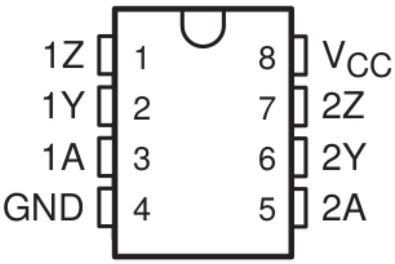
Q2

- Below is a figure of the TOE with the highlighted component:



- It has designator U15. The U is for integrated circuits. This is a part that's required for the "Grating" part of the PCB. The IC has a Texas Instruments logo and the numbers 75158. Searching this reveals that its a TI "SN75158 DUAL DIFFERENTIAL LINE DRIVER".

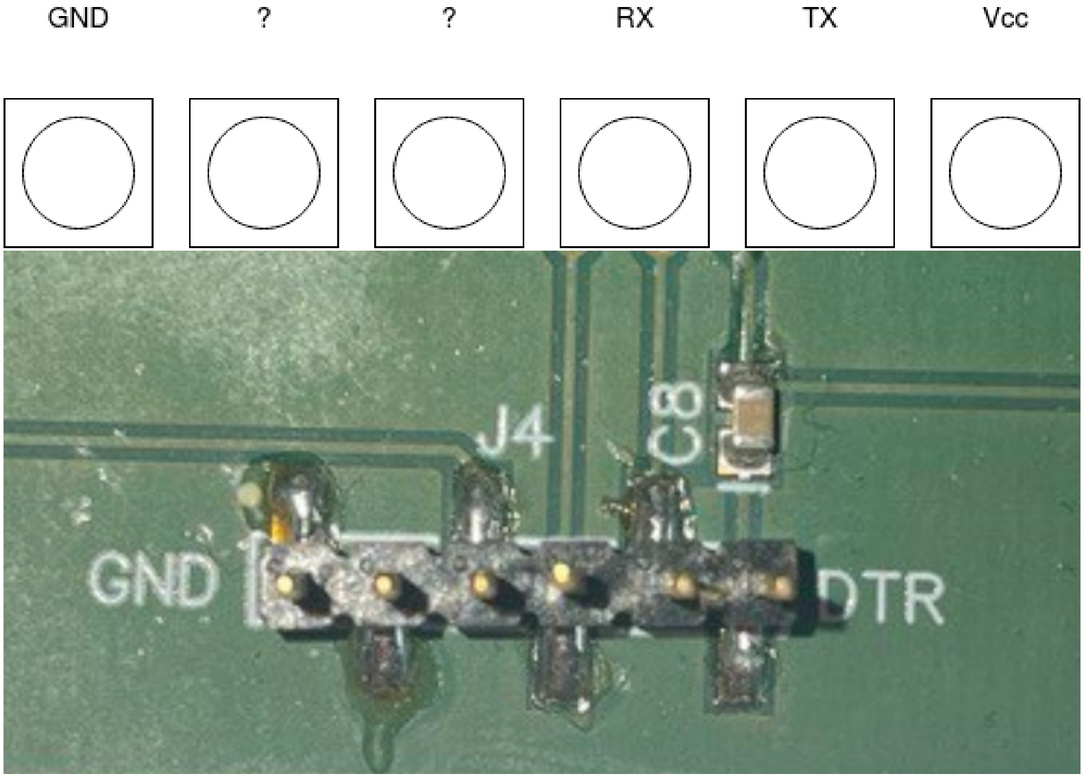
D, P, OR PS† PACKAGE
(TOP VIEW)



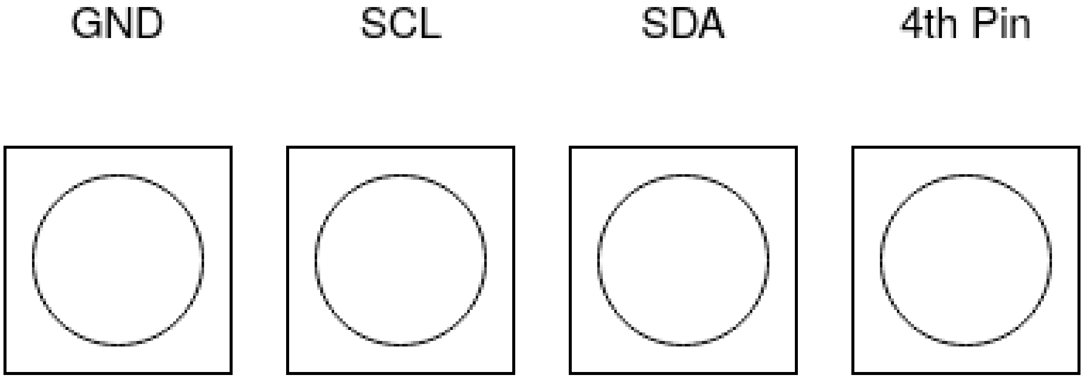
- The chip seems to split two inputs into two balanced lines. This is probably for carrying this signal over a longer distance with less distortion. Below is a diagram of the IC. 1A is split into 1Y (positive) and 1Z (negative). 2A is split into 2Y (positive) and 2Z (negative). GND is the ground and Vcc is the power needed to operate.

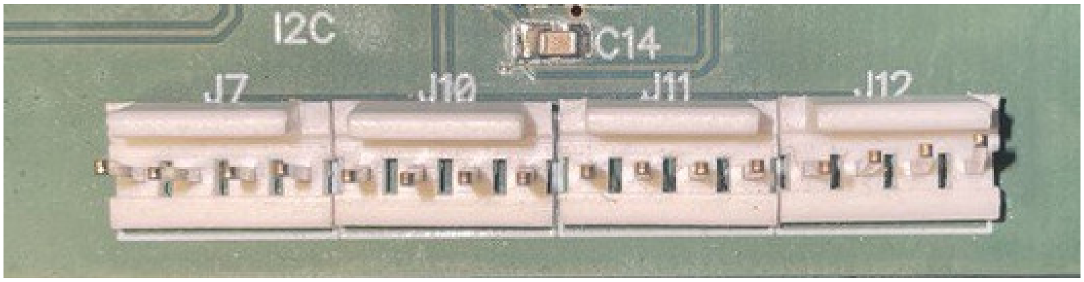
Q3

- Yes. See the pinout below. RX is connected to port 2, TX is connected to port 3, Vcc is connected to port 21



- There is no flash chip on the board.
- Yes! There are at least four I2C interfaces on the board. We are unsure what Pin 4 does, but each one is individually connected to an input on the CPU.




I2C
C14
J7
J10
J11
J12