Programming Report - Smart Environment

Forgotten Vegetables

Hilke van den Born, Anusha Autar, Daniël Ijtsma, Alessia Bertana, Rosalie van Elburg, Jannick Siderius

Initial Problem

For this project we had to do some research about smart environments. Thereafter we had to identify relevant problems. After doing that we came to the conclusion that in this day and age animal welfare is something that keeps on becoming more and more important. People are getting more conscious about the way animals are being treated. This has a lot of impact on the kind of animal-bases products that are being bought by consumers. If animals do not get treated well, consumers will not buy your product or even boycott it completely by protesting against it and sharing the fact that the animals are being mistreated with everyone, so more people will stand up against it.

The first problem that comes with this, is that it is hard for the farmers to ensure the health/welfare of the animals since it is hard to keep track of the health/welfare of the animals. At this moment the farmers mostly only monitor how much the food consumption of the animals is. The problem with this is that that does not really give a good or complete image about the health/welfare of the animals. There is not really a good way to monitor for instance whether the animals are feeling well or not, or whether they are healthy or not yet. That is something to worry about, because poor animal welfare impacts on animal production and reproduction and it can result in loss of market access. But the most important issue is that the animals are capable of feeling pain. They have a right to live a pleasurable life as well. So, for the sake of the animals, it is very important to be capable of monitoring their health/welfare.
There are already several organisations like “wakkerdier” and ”dierenbescherming“ that try to monitor it. However, in the technological field there is still a big gap.

Solution

The problem we chose, is a very recent day problem. There is a lot of debate about animal welfare. However, there is not much research on animal welfare, so that is the first thing we had to do, so we would know which aspects we had to pay attention to. As mentioned before there is still a big gap when it comes to the technological field to improve the animal health/welfare. We came up with the idea to solve this problem by for instance making some sort of an animal equivalent of a FitBit. That way it is a lot easier for the farmer to monitor whether the animals are healthy or not. We could use technology in many different ways. We could for example enable the farmer to turn his or her existing stables into smart environments that will help to boost the welfare of the animals.

To measure the health we could use a few parameters. The biggest and the most important parameters we want to measure are the movements of the animals, the temperature of the pigs and the noise levels in the stables. For tracking the health/welfare the farmer could use some kind of UI where he can see the amount of pigs, where they mostly are and what their health status is.

Method

Our idea is to make a PigBit, which tracks certain stats in every pig. This tracking is done by the PigBit itself that consists of an arduino with sensors that can sense the things we want to track. We want to track the following things:

- Movement
- Temperature
- Sound
- Light intensity

Because there is a lot of data the data will be collected by the hub. The hub

Movement will be tracked the following way:
We thought that the best way to track the movement of the pigs, is to put an RFID tag and an accelerometer on all of the FitBits(Ahmed, S. T., Mun, H.-S., Islam, Md. M., Yoe, H., & Yang, C.-J. (2015). Monitoring Activity for Recognition of Illness in Experimentally Infected Weaned Piglets Using Received Signal Strength Indication ZigBee-based Wireless Acceleration Sensor. *Asian-Australasian Journal of Animal Sciences*, *29*(1), 149–156. https://doi.org/10.5713/ajas.15.0221).The accelerometer is there to monitor the amount of daily movement the pigs are getting. For the RFID tag the system can see where every specific pig is because the RFID tags all have an individual ID. With this system it is thus very easy to see where which pig is. This information is for example useful to see in which part of the stable the pigs are. Is the pig in the outside part of the stable or is it inside?

Accelerometer + Gyroscope (6DOF sensor): MPU6050
(https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf)

RFID reader:	RDM6300 (http://www.mouser.com/catalog/specsheets/Seeed_113020002.pdf)
RFID tag:	EM4100 (http://www.priority1design.com.au/em4100_protocol.html)
	T5577

(https://store.dangerousthings.com/wp-content/uploads/doc_xEM_Atmel-9187-RFID-ATA5577C_Datasheet.pdf)

Temperature will be tracked the following way:
The temperature can be measured in multiple locations, in the stable as well as potentially for every individual pig. Stable temperature could be very easily measured at the hub or in more locations within the stable. A temperature measurement from the pig would be very difficult to achieve because you would always want to measure on the same spot, to not introduce a negative or positive bias to your measurements. We have thus concluded that contact-based measurement of the pigs temperature is not an option. We came up with the following solution to this problem: when pigs are feeding, we can measure their (skin)temperature by means of an IR-temperature sensor(Sellier, N., Guettier, E., & Staub, C. (2014). A Review of Methods to Measure Animal Body Temperature in Precision Farming. *American Journal of Agricultural Science and Technology*. https://doi.org/10.7726/ajast.2014.1008). This method required no physical contact and does not harm the animal in any way. With the temperature from every pig, analysis can be conducted to

see if the pig has a differing temperature from the normal, and action can be taken accordingly. By measuring with IR-temperature sensors it is possible to measure if the body temperature of pigs is around 39 degrees Celsius.

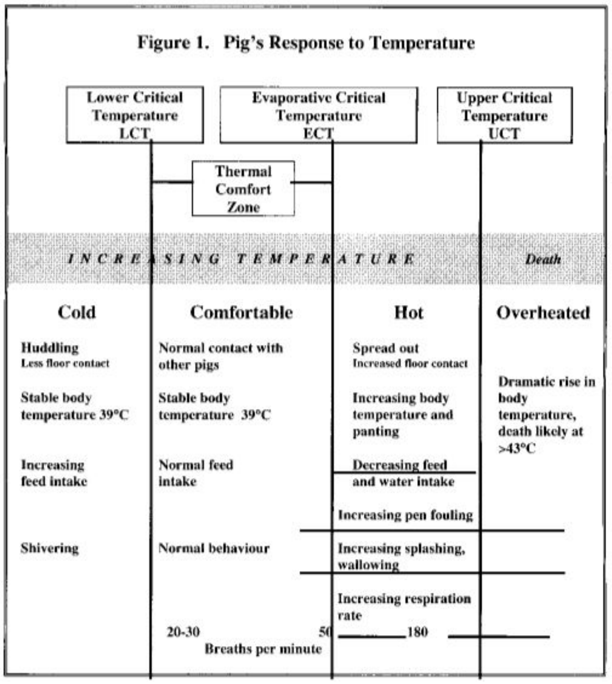


Figure 1. Pig's Response to Temperature

Lower Critical Temperature LCT
Evaporative Critical Temperature ECT
Upper Critical Temperature UCT
Thermal Comfort Zone
INCREASING TEMPERATURE
Death
Cold
Huddling
Less floor contact
Stable body temperature 39°C
Increasing feed intake
Shivering
Comfortable
Normal contact with other pigs
Stable body temperature 39°C
Normal feed intake
Normal behaviour
20-30
50
Breaths per minute
Hot
Spread out
Increased floor contact
Increasing body temperature and panting
Decreasing feed and water intake
Increasing pen fouling
Increasing splashing, wallowing
Increasing respiration rate
180
Overheated
Dramatic rise in body temperature, death likely at >43°C

Source:Lorschy, M. (1997). *Definitions of Ambient Temperature Requirements for Pigs:A Review*. Geraadpleegd van

https://fdocuments.in/document/temperature-requirements-for-pigs-prairie-temperature-requirements-for-pigs.html

Temperature sensor: LM35 (https://www.ti.com/lit/ds/symlink/lm35.pdf)
IR temperature sensor: MLX90614
(https://www.melexis.com/en/product/MLX90614/Digital-Plug-Play-Infrared-Thermometer-TO-Can)

Another way to measure the temperature is by implementing a temperature sensor(LM35) in the nodes that will be carried by the pigs. If the sensor is close enough to the skin or if it makes sensor-skin contact there will be a measurement of the surface temperature of a pig. To ensure good results it is necessary to have skin-sensor contact. The average skin temperature of a pig is around 35.6 degrees Celsius. (SOURCE: Soerensen, D. D., & Pedersen, L. J. (2015). Infrared skin temperature measurements for monitoring health in pigs: a review. *Acta Veterinaria Scandinavica*, *57*(1). https://doi.org/10.1186/s13028-015-0094-2)

Next to the body temperature we also measure the stable temperature. In order to protect the welfare of the pigs it has to be around 23 degrees celsius. (SOURCE: da Massabie, P., & Granier, R. (2001). *Effect of Air movement and ambient temperature on the zootechnical performance and behaviour of growing-finishing pigs*. Geraadpleegd op 13 december 2019, van https://elibrary.asabe.org/abstract.asp?aid=3536)

Sound will be tracked the following way:

Sound tracking is a fairly easy thing to measure. Just a sensor that can detect sound levels. One thing we can think about here, is whether we want one sound sensor in the stable that tracks all sound in the entire stable, or if we want a sound sensor in every PigBit to be able to track more accurately which pig is making the noise. In order to keep pigs stress-free it is necessary to keep the noise level below 85dB. (SOURCE: Mul, M., Vermeij, I., Hindle, V., & Spoolder, H. (2010). EU-Welfare legislation on pigs. , 17.)

Microphone: WM-61A
(https://components101.com/sites/default/files/component_datasheet/Electret%20Condenser%20Microphone.pdf)

Light intensity will be tracked the following way:
To measure light intensity we do not need a lot of material. All we need is a light dependent resistor(LDR). The value of this resistor depends on the light intensity, because of this we can use the output of this resistor as a measurement for the light intensity. A pig has to be exposed to be at least 40 lux of light for eight hours a day. (SOURCE: Mul, M., Vermeij, I., Hindle, V., & Spoolder, H. (2010). EU-Welfare legislation on pigs. , 17.)

LDR: GL5537 LDR

One other thing that we will implement in the PigBit is a little LED with two or three settings: either red, blue or green light. Each light has another meaning.
When we implement the three color option the light on one of the PigBit is red, the farmer knows that something is wrong with this pig, and he can go check it out. If the light is green, the farmer knows that everything is good with that pig. The farmer itself can set the LED of a specific pig to blue, when he wants to find that pig. Whether the LED of a pig is red or green, depending on the data that the sensors are collecting. If for example, a pig has a way to high temperature or if the sensors have picked up that a pig has not reached enough steps, the LED of that pig will turn red.

When we implement the two-color option the LED can turn green or red. When a LED turns green this tells the farmer that a specific node is connected to the mesh network and when the LED turns red it means that the specific node is not connected to the mesh network. By using this method the farmer can get an insight into how his network is working and he can see directly if a node is connected or not.

RGB Led (common cathode):
WS2812b addressable led (https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf)

The PigBit will need a rather tough casing as pigs are a pretty rowdy bunch of creatures that can easily destroy their own PigBit or that of other pigs, certainly when it is hanging around their necks (which is the case), because that’s a location that other pigs can easily bump into. What we have to consider here is that we have to make it sturdy, but not too big since you don’t want all of your pigs to walk around with a brick sized lump on their necks.

The data from the sensors will be tracked around the clock. What the pigs are doing and how they are behaving is constantly changing, so if you want accurate measurements, you can’t turn the PigBit off during certain parts of the day, because you might miss important information. Disadvantage of this is that we will collect a lot of data, that needs to be processed by a computer that can handle that amount. Therefore, we have to test in an application of the prototype what a good data collection rate would be, eg. 1 measurement / min or 1 measurement / 5 seconds.

Results

In chapter 7 of the original documentation, the methodology, we mentioned what we could possibly need to make the PigBit work. After some revision we decided to go for the following strategy:

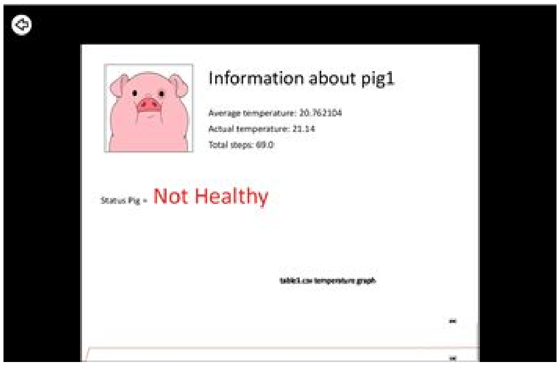
- By putting an RFID tag and an accelerometer on all of the PigBit, the movement and the location will be tracked.
- By using an IR-temperature sensor, the (skin)temperature of every individual pig will be measured. Besides that the temperature of multiple locations in the stable will be measured as well.
- By using a microphone, the sound level will be measured. The sound will be measured in every PigBit to track which pig is making the noise.
- LEDs are supposed to turn red or green depending on whether the sensors are connected or not.

We tested this strategy by letting 3 people walk around and act like pigs. This way we were able to see whether or not the movement and location were being tracked. Besides that, we let one person walk around with the RFID tag and one person with the hub. The “pigs” walked around for a couple of hours and every once in a while, they passed by the RFID tag. Via the hub we checked whether or not there was incoming data.

After testing our strategy, we came to the conclusion that the movement and location tracker was working the way we anticipated it would be. However, even though we did collect data from the temperature sensor, we were not sure whether the data was as accurate as we wanted it to be. Via the hub, we could see changes in temperature. The measuring of the sound level did not work out the way we wanted it to be either. It was malfunctioning for no apparent reason. The light sensor did work how we anticipated it would be.

On the next few pages, you can see the test data we collected.

In the UI we show the user or farmer the results of our sensor measurements (so the temperatures, sound, light, and steps). We integrated an algorithm to determine whether a pig is healthy or not. Among others we display a graph which shows the temperature of the pig over time. We display a graph for the temperature of the stable as well:



Information about pig1

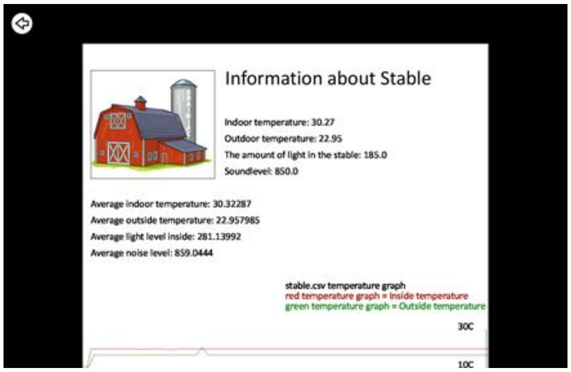
Average temperature: 20.762104

Actual temperature: 21.14

Total steps: 69.0

Status Pig = Not Healthy

table1.csv temperature graph



Information about Stable

Indoor temperature: 30.27

Outdoor temperature: 22.95

The amount of light in the stable: 185.0

Soundlevel: 850.0

Average indoor temperature: 30.32287

Average outside temperature: 22.957985

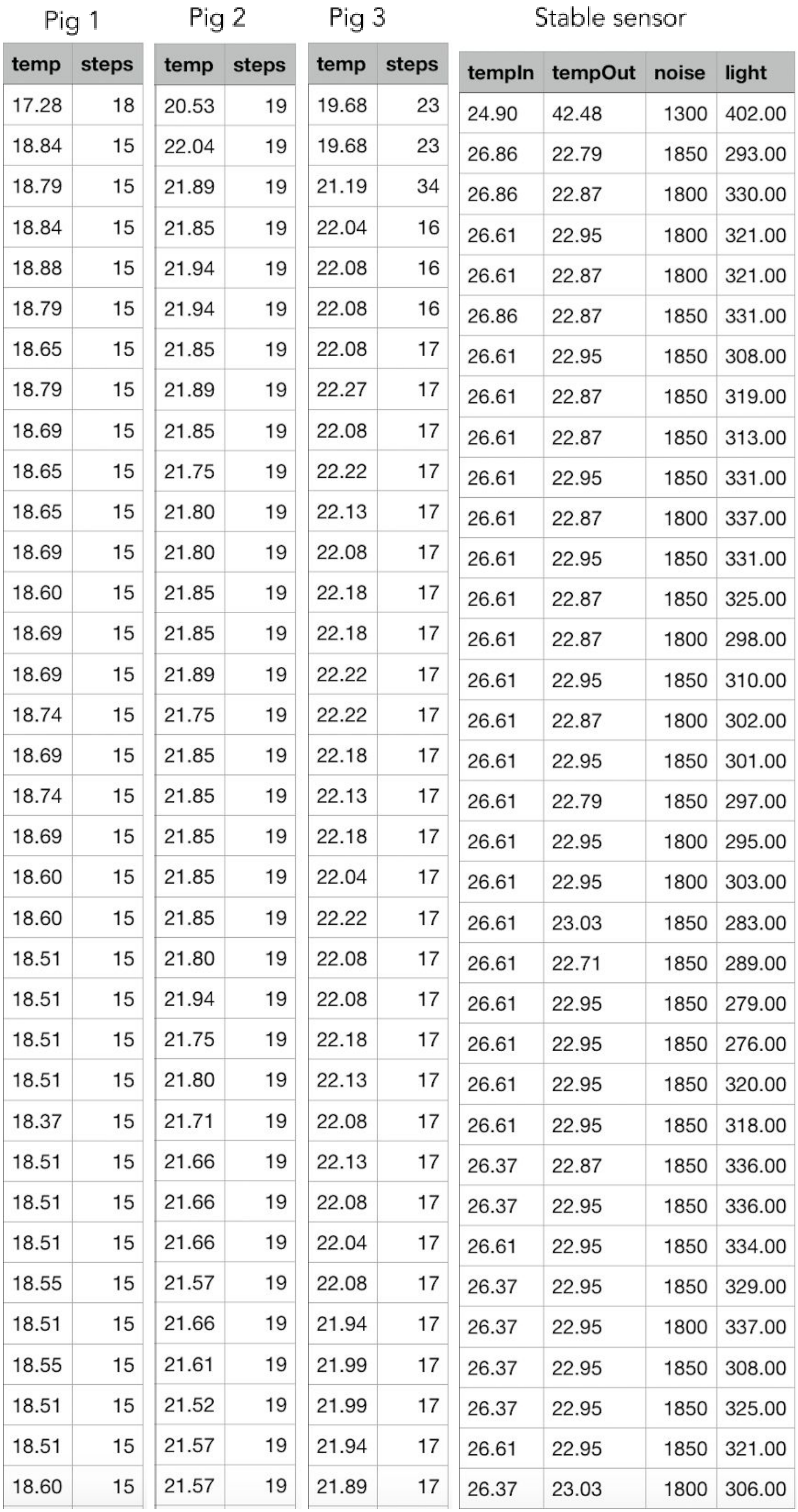
Average light level inside: 281.13992

Average noise level: 859.0444

stable.csv temperature graph

red temperature graph = Inside temperature

green temperature graph = Outside temperature



Pig 1

temp	steps
17.28	18
18.84	15
18.79	15
18.84	15
18.88	15
18.79	15
18.65	15
18.79	15
18.69	15
18.65	15
18.65	15
18.69	15
18.60	15
18.69	15
18.69	15
18.74	15
18.69	15
18.74	15
18.69	15
18.60	15
18.60	15
18.51	15
18.51	15
18.51	15
18.51	15
18.37	15
18.51	15
18.51	15
18.51	15
18.55	15
18.51	15
18.55	15
18.51	15
18.51	15
18.60	15

Pig 2

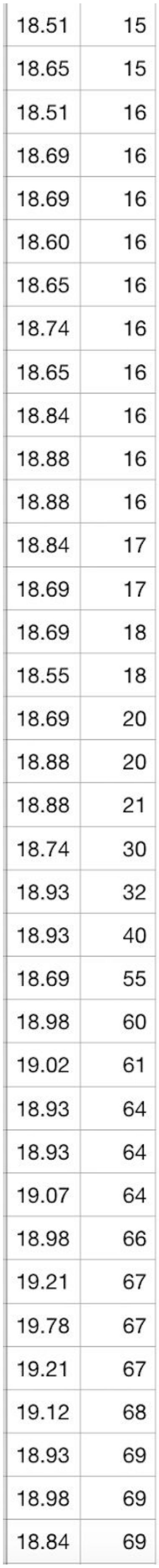
temp	steps
20.53	19
22.04	19
21.89	19
21.85	19
21.94	19
21.94	19
21.85	19
21.89	19
21.85	19
21.75	19
21.80	19
21.80	19
21.85	19
21.85	19
21.89	19
21.75	19
21.85	19
21.85	19
21.85	19
21.85	19
21.85	19
21.80	19
21.94	19
21.75	19
21.80	19
21.71	19
21.66	19
21.66	19
21.66	19
21.57	19
21.66	19
21.61	19
21.52	19
21.57	19
21.57	19

Pig 3

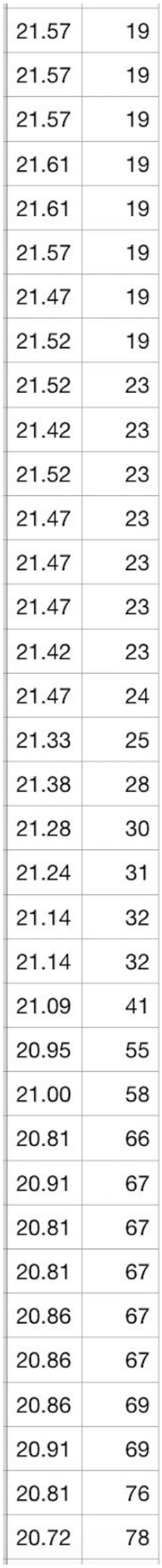
temp	steps
19.68	23
19.68	23
21.19	34
22.04	16
22.08	16
22.08	16
22.08	17
22.27	17
22.08	17
22.22	17
22.13	17
22.08	17
22.18	17
22.18	17
22.22	17
22.22	17
22.18	17
22.13	17
22.18	17
22.04	17
22.22	17
22.08	17
22.08	17
22.18	17
22.13	17
22.08	17
22.13	17
22.08	17
22.04	17
22.08	17
21.94	17
21.99	17
21.99	17
21.94	17
21.89	17

Stable sensor

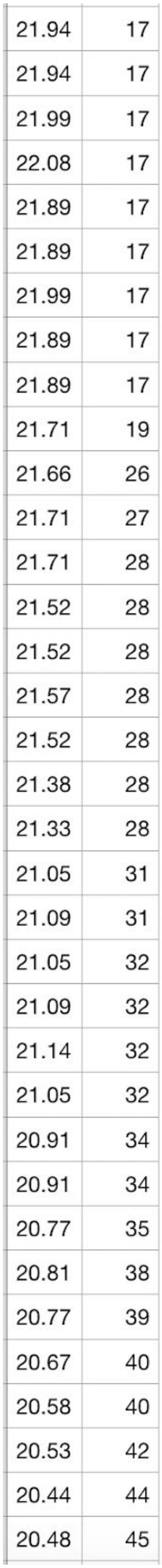
tempIn	tempOut	noise	light
24.90	42.48	1300	402.00
26.86	22.79	1850	293.00
26.86	22.87	1800	330.00
26.61	22.95	1800	321.00
26.61	22.87	1800	321.00
26.86	22.87	1850	331.00
26.61	22.95	1850	308.00
26.61	22.87	1850	319.00
26.61	22.87	1850	313.00
26.61	22.95	1850	331.00
26.61	22.87	1800	337.00
26.61	22.95	1850	331.00
26.61	22.87	1850	325.00
26.61	22.87	1800	298.00
26.61	22.95	1850	310.00
26.61	22.87	1800	302.00
26.61	22.95	1850	301.00
26.61	22.79	1850	297.00
26.61	22.95	1800	295.00
26.61	22.95	1800	303.00
26.61	23.03	1850	283.00
26.61	22.71	1850	289.00
26.61	22.95	1850	279.00
26.61	22.95	1850	276.00
26.61	22.95	1850	320.00
26.61	22.95	1850	318.00
26.37	22.87	1850	336.00
26.37	22.95	1850	336.00
26.61	22.95	1850	334.00
26.37	22.95	1850	329.00
26.37	22.95	1800	337.00
26.37	22.95	1850	308.00
26.37	22.95	1850	325.00
26.61	22.95	1850	321.00
26.37	23.03	1800	306.00



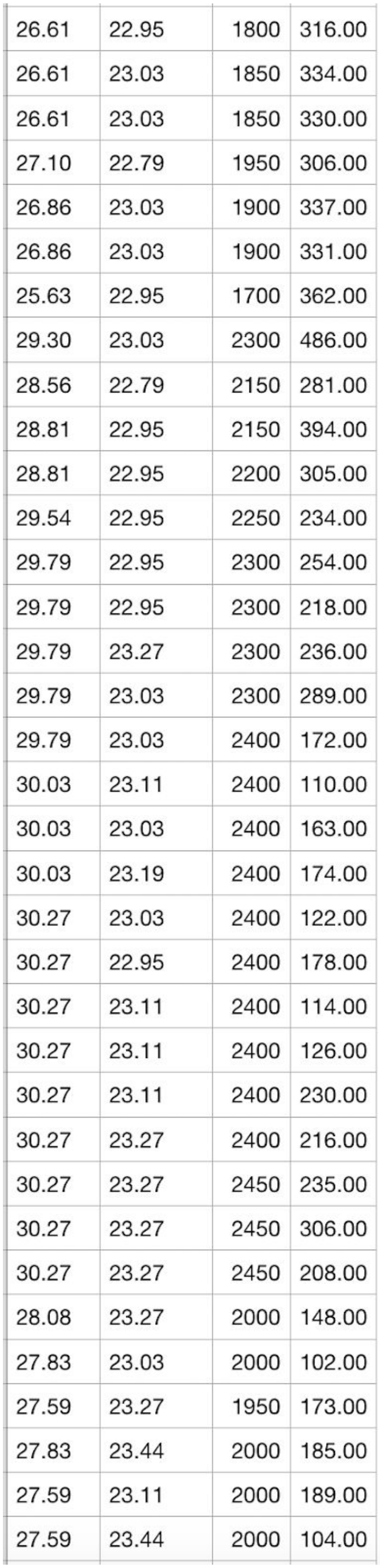
18.51	15
18.65	15
18.51	16
18.69	16
18.69	16
18.60	16
18.65	16
18.74	16
18.65	16
18.84	16
18.88	16
18.88	16
18.84	17
18.69	17
18.69	18
18.55	18
18.69	20
18.88	20
18.88	21
18.74	30
18.93	32
18.93	40
18.69	55
18.98	60
19.02	61
18.93	64
18.93	64
19.07	64
18.98	66
19.21	67
19.78	67
19.21	67
19.12	68
18.93	69
18.98	69
18.84	69



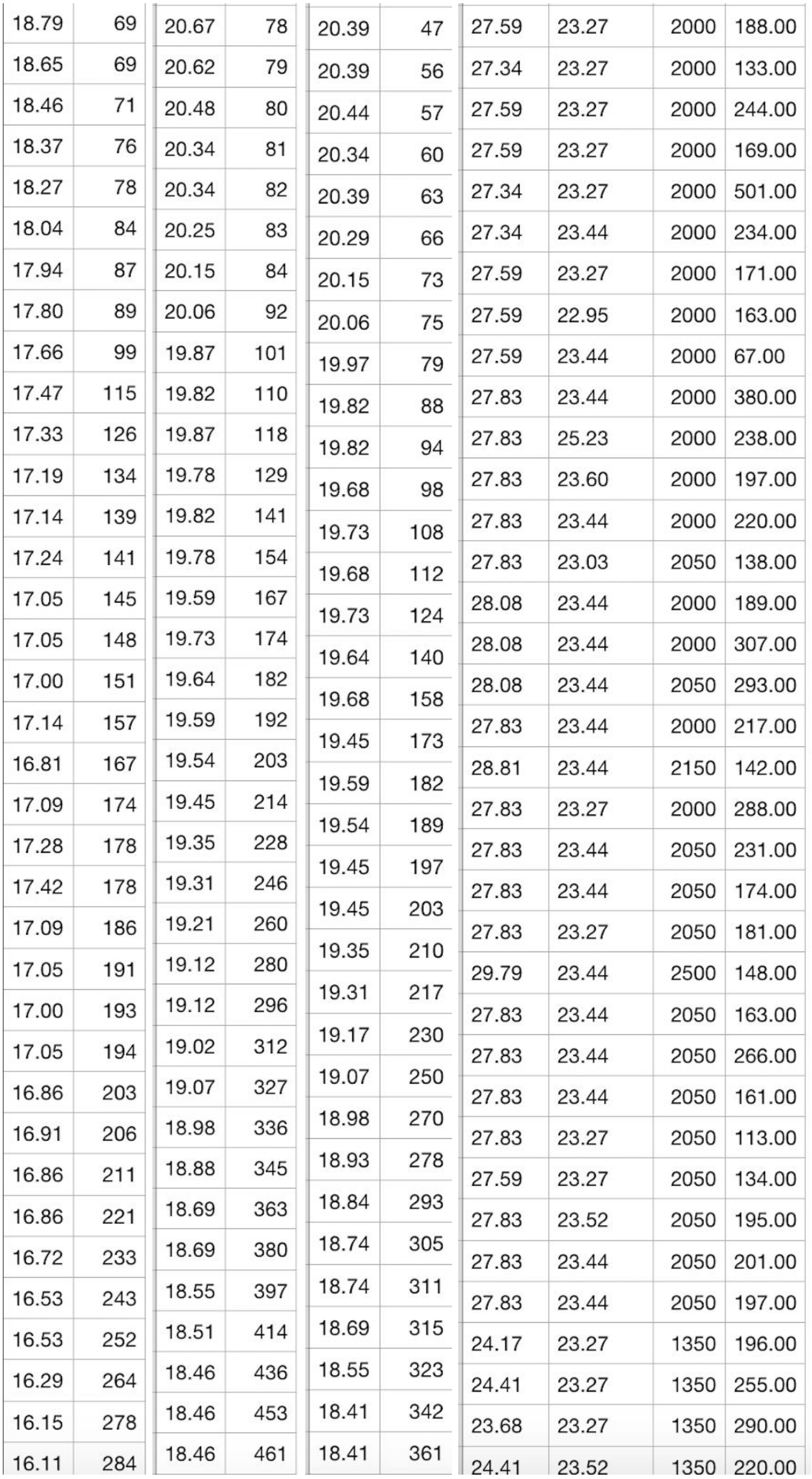
21.57	19
21.57	19
21.57	19
21.61	19
21.61	19
21.57	19
21.47	19
21.52	19
21.52	23
21.42	23
21.52	23
21.47	23
21.47	23
21.47	23
21.42	23
21.47	24
21.33	25
21.38	28
21.28	30
21.24	31
21.14	32
21.14	32
21.09	41
20.95	55
21.00	58
20.81	66
20.91	67
20.81	67
20.81	67
20.86	67
20.86	67
20.86	69
20.91	69
20.81	76
20.72	78



21.94	17
21.94	17
21.99	17
22.08	17
21.89	17
21.89	17
21.99	17
21.89	17
21.89	17
21.71	19
21.66	26
21.71	27
21.71	28
21.52	28
21.52	28
21.57	28
21.52	28
21.38	28
21.33	28
21.05	31
21.09	31
21.05	32
21.09	32
21.14	32
21.05	32
20.91	34
20.91	34
20.77	35
20.81	38
20.77	39
20.67	40
20.58	40
20.53	42
20.44	44
20.48	45



26.61	22.95	1800	316.00
26.61	23.03	1850	334.00
26.61	23.03	1850	330.00
27.10	22.79	1950	306.00
26.86	23.03	1900	337.00
26.86	23.03	1900	331.00
25.63	22.95	1700	362.00
29.30	23.03	2300	486.00
28.56	22.79	2150	281.00
28.81	22.95	2150	394.00
28.81	22.95	2200	305.00
29.54	22.95	2250	234.00
29.79	22.95	2300	254.00
29.79	22.95	2300	218.00
29.79	23.27	2300	236.00
29.79	23.03	2300	289.00
29.79	23.03	2400	172.00
30.03	23.11	2400	110.00
30.03	23.03	2400	163.00
30.03	23.19	2400	174.00
30.27	23.03	2400	122.00
30.27	22.95	2400	178.00
30.27	23.11	2400	114.00
30.27	23.11	2400	126.00
30.27	23.11	2400	230.00
30.27	23.27	2400	216.00
30.27	23.27	2450	235.00
30.27	23.27	2450	306.00
30.27	23.27	2450	208.00
28.08	23.27	2000	148.00
27.83	23.03	2000	102.00
27.59	23.27	1950	173.00
27.83	23.44	2000	185.00
27.59	23.11	2000	189.00
27.59	23.44	2000	104.00

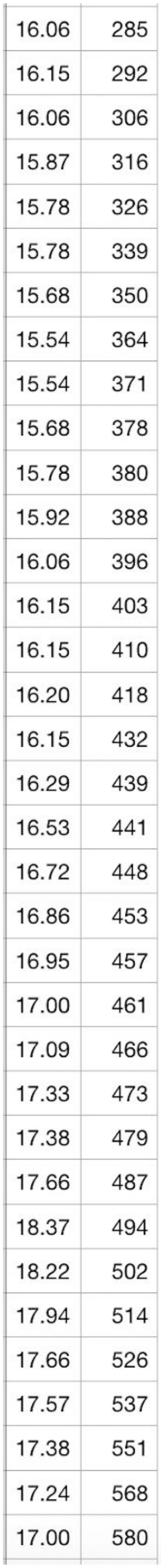


18.79	69
18.65	69
18.46	71
18.37	76
18.27	78
18.04	84
17.94	87
17.80	89
17.66	99
17.47	115
17.33	126
17.19	134
17.14	139
17.24	141
17.05	145
17.05	148
17.00	151
17.14	157
16.81	167
17.09	174
17.28	178
17.42	178
17.09	186
17.05	191
17.00	193
17.05	194
16.86	203
16.91	206
16.86	211
16.86	221
16.72	233
16.53	243
16.53	252
16.29	264
16.15	278
16.11	284

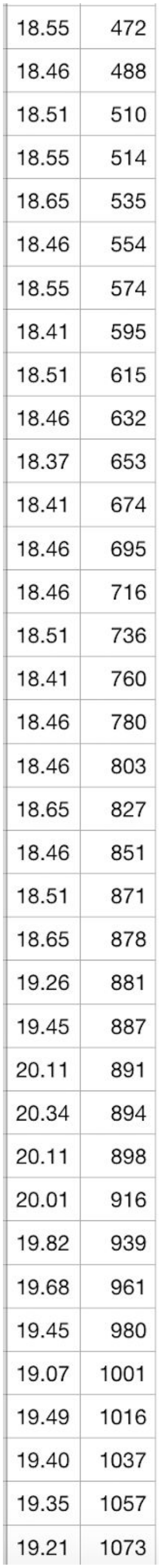
20.67	78
20.62	79
20.48	80
20.34	81
20.34	82
20.25	83
20.15	84
20.06	92
19.87	101
19.82	110
19.87	118
19.78	129
19.82	141
19.78	154
19.59	167
19.73	174
19.64	182
19.59	192
19.54	203
19.45	214
19.35	228
19.31	246
19.21	260
19.12	280
19.12	296
19.02	312
19.07	327
18.98	336
18.88	345
18.69	363
18.69	380
18.55	397
18.51	414
18.46	436
18.46	453
18.46	461

20.39	47
20.39	56
20.44	57
20.34	60
20.39	63
20.29	66
20.15	73
20.06	75
19.97	79
19.82	88
19.82	94
19.68	98
19.73	108
19.68	112
19.73	124
19.64	140
19.68	158
19.45	173
19.59	182
19.54	189
19.45	197
19.45	203
19.35	210
19.31	217
19.17	230
19.07	250
18.98	270
18.93	278
18.84	293
18.74	305
18.74	311
18.69	315
18.55	323
18.41	342
18.41	361

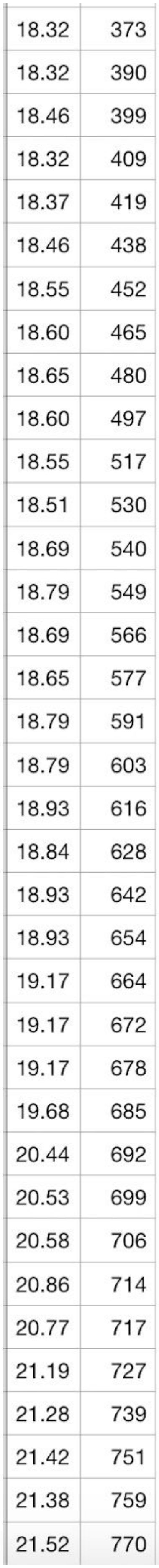
27.59	23.27	2000	188.00
27.34	23.27	2000	133.00
27.59	23.27	2000	244.00
27.59	23.27	2000	169.00
27.34	23.27	2000	501.00
27.34	23.44	2000	234.00
27.59	23.27	2000	171.00
27.59	22.95	2000	163.00
27.59	23.44	2000	67.00
27.83	23.44	2000	380.00
27.83	25.23	2000	238.00
27.83	23.60	2000	197.00
27.83	23.44	2000	220.00
27.83	23.03	2050	138.00
28.08	23.44	2000	189.00
28.08	23.44	2000	307.00
28.08	23.44	2050	293.00
27.83	23.44	2000	217.00
28.81	23.44	2150	142.00
27.83	23.27	2000	288.00
27.83	23.44	2050	231.00
27.83	23.44	2050	174.00
27.83	23.27	2050	181.00
29.79	23.44	2500	148.00
27.83	23.44	2050	163.00
27.83	23.44	2050	266.00
27.83	23.44	2050	161.00
27.83	23.27	2050	113.00
27.59	23.27	2050	134.00
27.83	23.52	2050	195.00
27.83	23.44	2050	201.00
27.83	23.44	2050	197.00
24.17	23.27	1350	196.00
24.41	23.27	1350	255.00
23.68	23.27	1350	290.00
24.41	23.52	1350	220.00



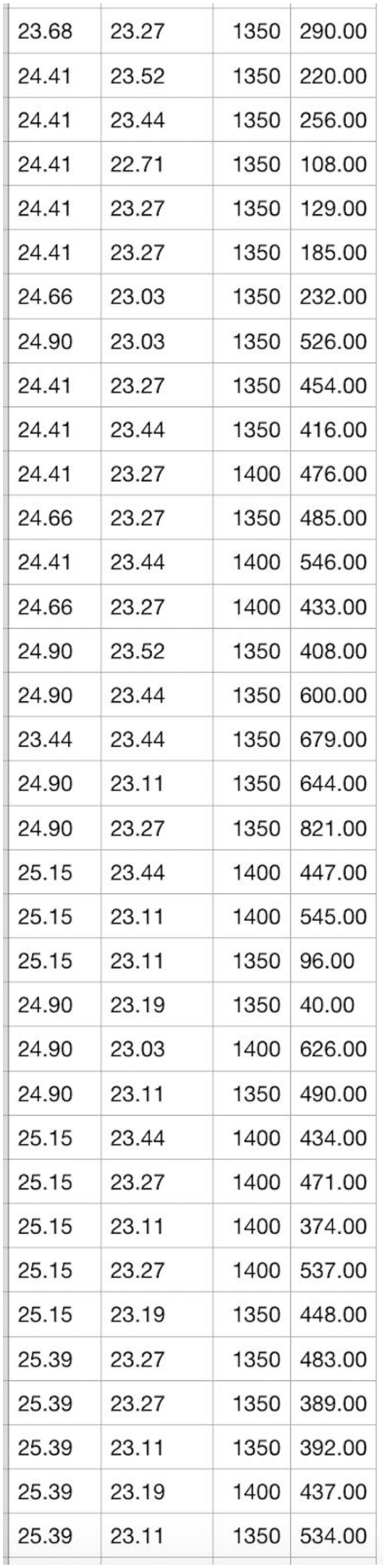
16.06	285
16.15	292
16.06	306
15.87	316
15.78	326
15.78	339
15.68	350
15.54	364
15.54	371
15.68	378
15.78	380
15.92	388
16.06	396
16.15	403
16.15	410
16.20	418
16.15	432
16.29	439
16.53	441
16.72	448
16.86	453
16.95	457
17.00	461
17.09	466
17.33	473
17.38	479
17.66	487
18.37	494
18.22	502
17.94	514
17.66	526
17.57	537
17.38	551
17.24	568
17.00	580



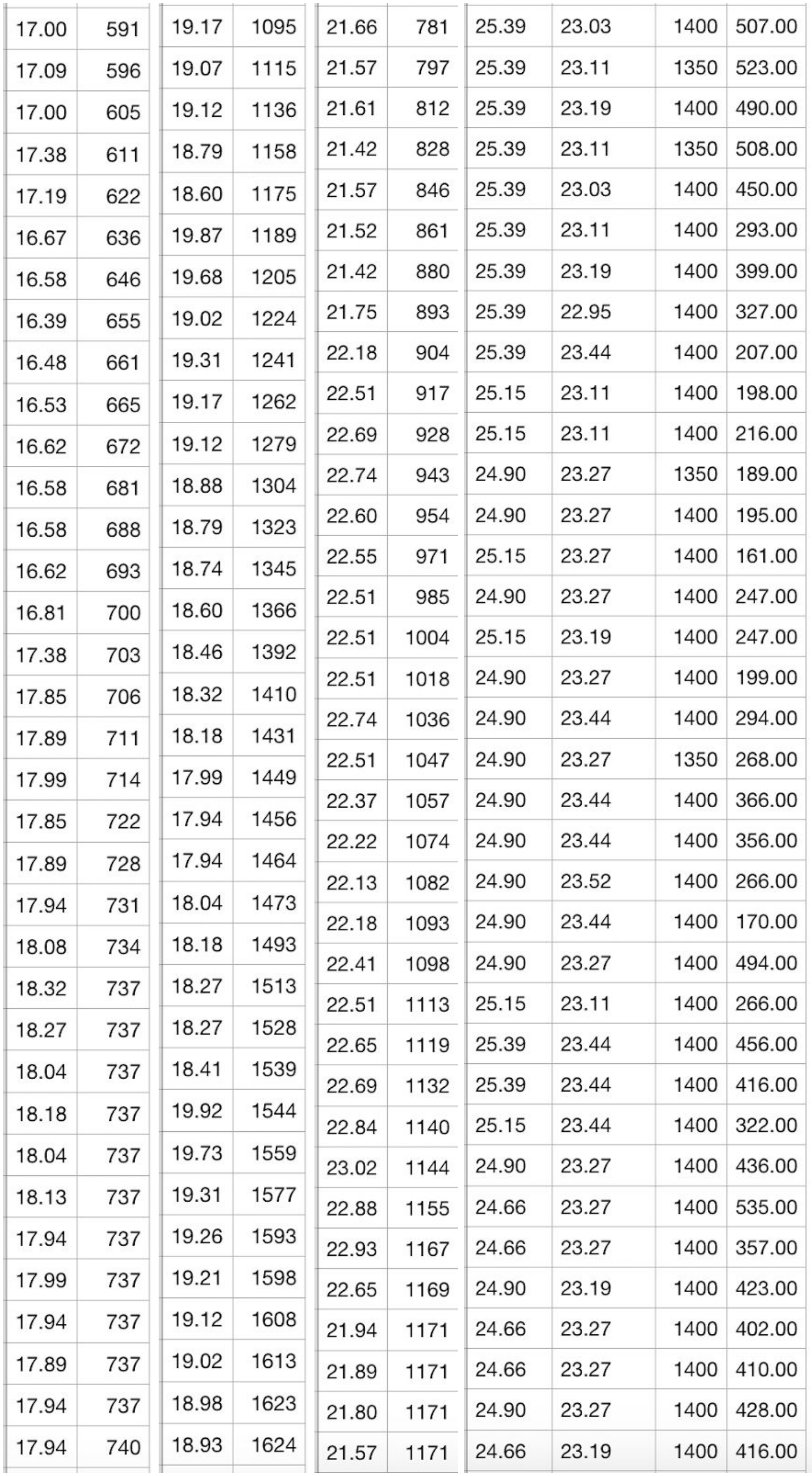
18.55	472
18.46	488
18.51	510
18.55	514
18.65	535
18.46	554
18.55	574
18.41	595
18.51	615
18.46	632
18.37	653
18.41	674
18.46	695
18.46	716
18.51	736
18.41	760
18.46	780
18.46	803
18.65	827
18.46	851
18.51	871
18.65	878
19.26	881
19.45	887
20.11	891
20.34	894
20.11	898
20.01	916
19.82	939
19.68	961
19.45	980
19.07	1001
19.49	1016
19.40	1037
19.35	1057
19.21	1073



18.32	373
18.32	390
18.46	399
18.32	409
18.37	419
18.46	438
18.55	452
18.60	465
18.65	480
18.60	497
18.55	517
18.51	530
18.69	540
18.79	549
18.69	566
18.65	577
18.79	591
18.79	603
18.93	616
18.84	628
18.93	642
18.93	654
19.17	664
19.17	672
19.17	678
19.68	685
20.44	692
20.53	699
20.58	706
20.86	714
20.77	717
21.19	727
21.28	739
21.42	751
21.38	759
21.52	770



23.68	23.27	1350	290.00
24.41	23.52	1350	220.00
24.41	23.44	1350	256.00
24.41	22.71	1350	108.00
24.41	23.27	1350	129.00
24.41	23.27	1350	185.00
24.66	23.03	1350	232.00
24.90	23.03	1350	526.00
24.41	23.27	1350	454.00
24.41	23.44	1350	416.00
24.41	23.27	1400	476.00
24.66	23.27	1350	485.00
24.41	23.44	1400	546.00
24.66	23.27	1400	433.00
24.90	23.52	1350	408.00
24.90	23.44	1350	600.00
23.44	23.44	1350	679.00
24.90	23.11	1350	644.00
24.90	23.27	1350	821.00
25.15	23.44	1400	447.00
25.15	23.11	1400	545.00
25.15	23.11	1350	96.00
24.90	23.19	1350	40.00
24.90	23.03	1400	626.00
24.90	23.11	1350	490.00
25.15	23.44	1400	434.00
25.15	23.27	1400	471.00
25.15	23.11	1400	374.00
25.15	23.27	1400	537.00
25.15	23.19	1350	448.00
25.39	23.27	1350	483.00
25.39	23.27	1350	389.00
25.39	23.11	1350	392.00
25.39	23.19	1400	437.00
25.39	23.11	1350	534.00

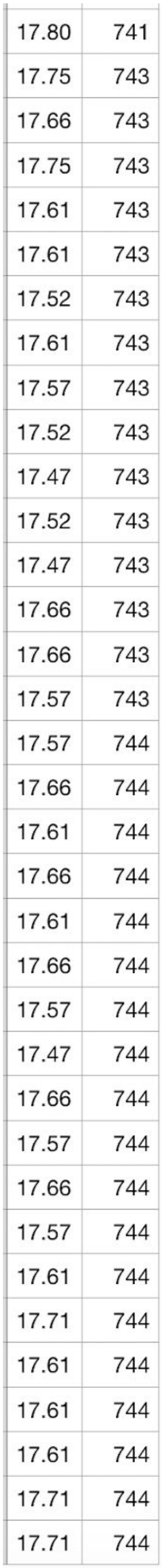


17.00	591
17.09	596
17.00	605
17.38	611
17.19	622
16.67	636
16.58	646
16.39	655
16.48	661
16.53	665
16.62	672
16.58	681
16.58	688
16.62	693
16.81	700
17.38	703
17.85	706
17.89	711
17.99	714
17.85	722
17.89	728
17.94	731
18.08	734
18.32	737
18.27	737
18.04	737
18.18	737
18.04	737
18.13	737
17.94	737
17.99	737
17.94	737
17.89	737
17.94	737
17.94	740

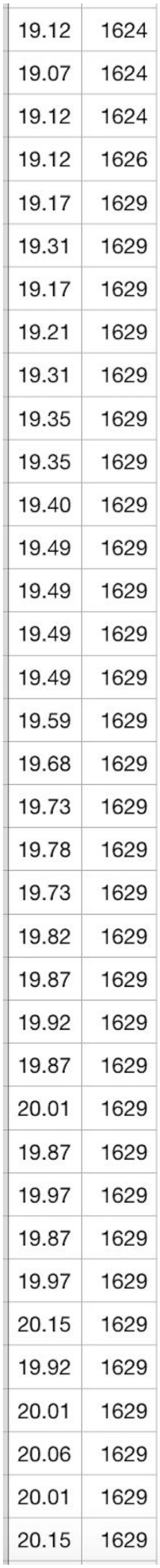
19.17	1095
19.07	1115
19.12	1136
18.79	1158
18.60	1175
19.87	1189
19.68	1205
19.02	1224
19.31	1241
19.17	1262
19.12	1279
18.88	1304
18.79	1323
18.74	1345
18.60	1366
18.46	1392
18.32	1410
18.18	1431
17.99	1449
17.94	1456
17.94	1464
18.04	1473
18.18	1493
18.27	1513
18.27	1528
18.41	1539
19.92	1544
19.73	1559
19.31	1577
19.26	1593
19.21	1598
19.12	1608
19.02	1613
18.98	1623
18.93	1624

21.66	781
21.57	797
21.61	812
21.42	828
21.57	846
21.52	861
21.42	880
21.75	893
22.18	904
22.51	917
22.69	928
22.74	943
22.60	954
22.55	971
22.51	985
22.51	1004
22.51	1018
22.74	1036
22.51	1047
22.37	1057
22.22	1074
22.13	1082
22.18	1093
22.41	1098
22.51	1113
22.65	1119
22.69	1132
22.84	1140
23.02	1144
22.88	1155
22.93	1167
22.65	1169
21.94	1171
21.89	1171
21.80	1171
21.57	1171

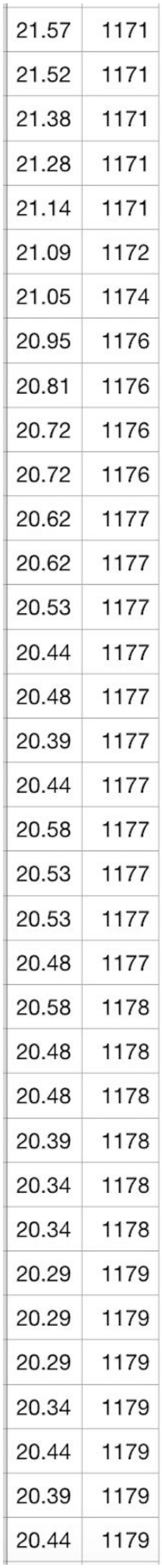
25.39	23.03	1400	507.00
25.39	23.11	1350	523.00
25.39	23.19	1400	490.00
25.39	23.11	1350	508.00
25.39	23.03	1400	450.00
25.39	23.11	1400	293.00
25.39	23.19	1400	399.00
25.39	22.95	1400	327.00
25.39	23.44	1400	207.00
25.15	23.11	1400	198.00
25.15	23.11	1400	216.00
24.90	23.27	1350	189.00
24.90	23.27	1400	195.00
25.15	23.27	1400	161.00
24.90	23.27	1400	247.00
25.15	23.19	1400	247.00
24.90	23.27	1400	199.00
24.90	23.44	1400	294.00
24.90	23.27	1350	268.00
24.90	23.44	1400	366.00
24.90	23.44	1400	356.00
24.90	23.52	1400	266.00
24.90	23.44	1400	170.00
24.90	23.27	1400	494.00
25.15	23.11	1400	266.00
25.39	23.44	1400	456.00
25.39	23.44	1400	416.00
25.15	23.44	1400	322.00
24.90	23.27	1400	436.00
24.66	23.27	1400	535.00
24.66	23.27	1400	357.00
24.90	23.19	1400	423.00
24.66	23.27	1400	402.00
24.66	23.27	1400	410.00
24.90	23.27	1400	428.00
24.66	23.19	1400	416.00



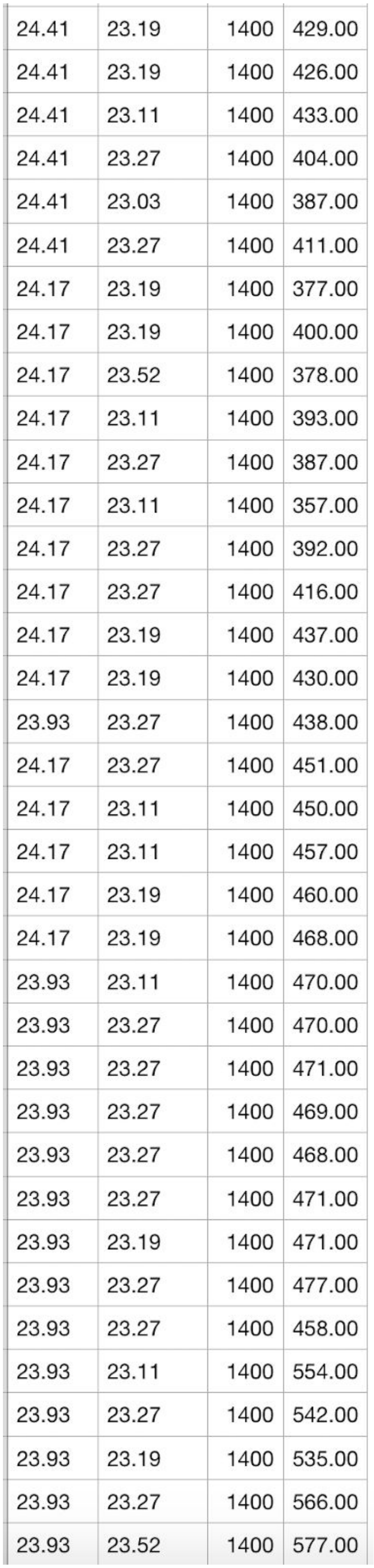
17.80	741
17.75	743
17.66	743
17.75	743
17.61	743
17.61	743
17.52	743
17.61	743
17.57	743
17.52	743
17.47	743
17.52	743
17.47	743
17.66	743
17.66	743
17.57	743
17.57	744
17.66	744
17.61	744
17.66	744
17.61	744
17.66	744
17.57	744
17.47	744
17.66	744
17.57	744
17.66	744
17.57	744
17.61	744
17.71	744
17.61	744
17.61	744
17.61	744
17.71	744
17.71	744



19.12	1624
19.07	1624
19.12	1624
19.12	1626
19.17	1629
19.31	1629
19.17	1629
19.21	1629
19.31	1629
19.35	1629
19.35	1629
19.40	1629
19.49	1629
19.49	1629
19.49	1629
19.49	1629
19.59	1629
19.68	1629
19.73	1629
19.78	1629
19.73	1629
19.82	1629
19.87	1629
19.92	1629
19.87	1629
20.01	1629
19.87	1629
19.97	1629
19.87	1629
19.97	1629
20.15	1629
19.92	1629
20.01	1629
20.06	1629
20.01	1629
20.15	1629

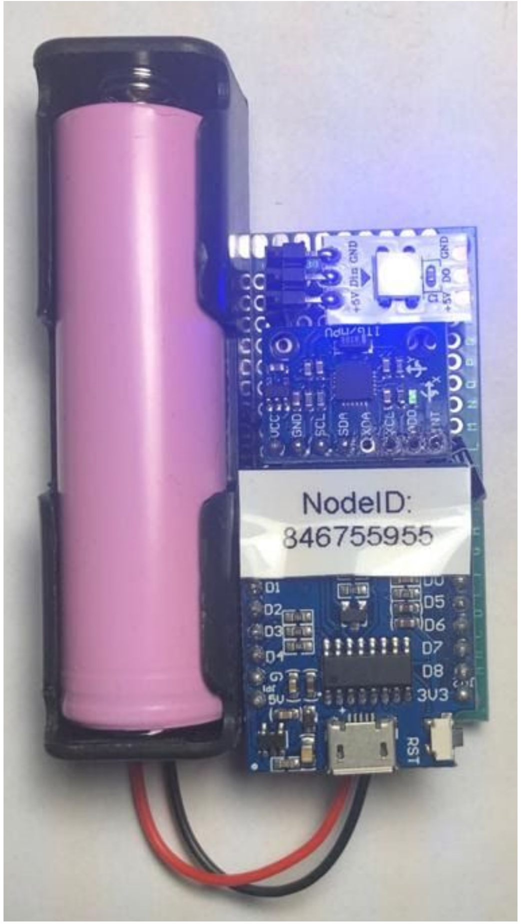


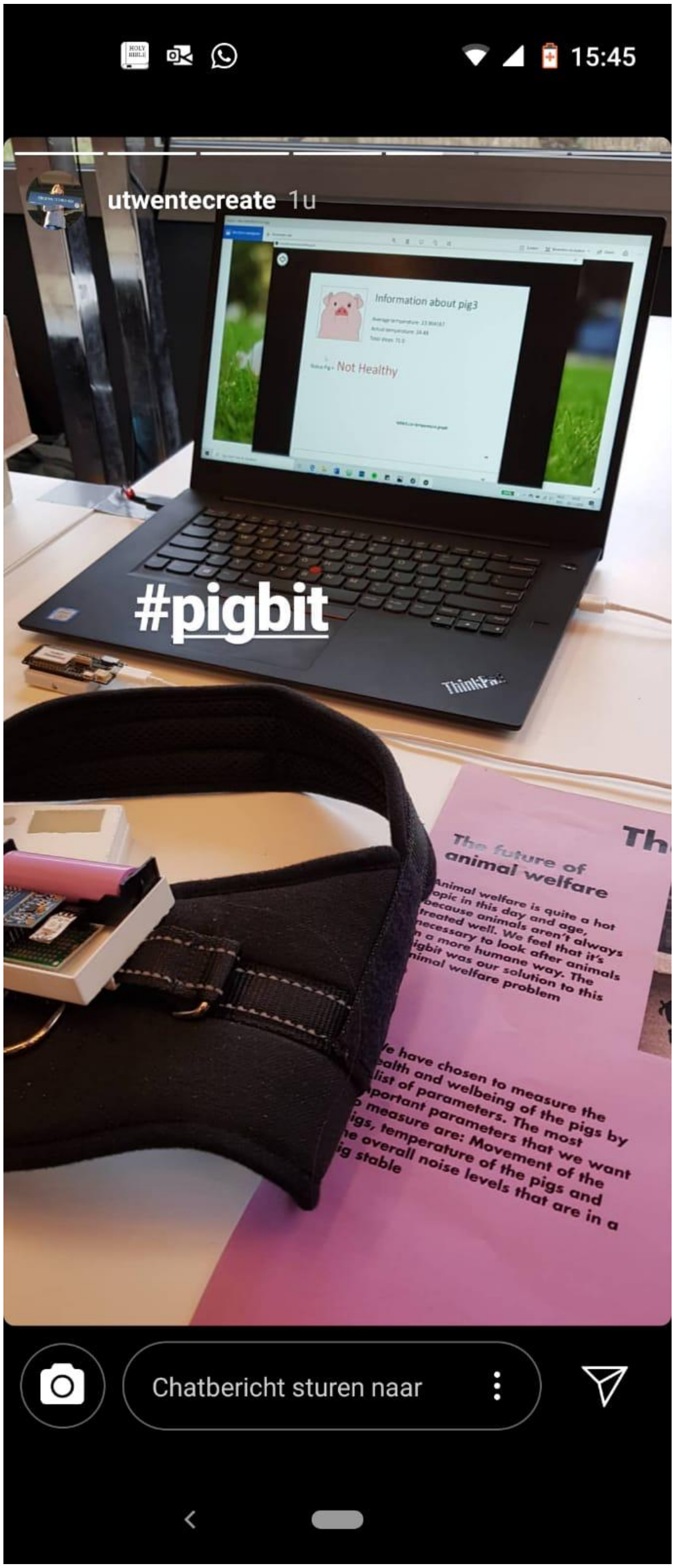
21.57	1171
21.52	1171
21.38	1171
21.28	1171
21.14	1171
21.09	1172
21.05	1174
20.95	1176
20.81	1176
20.72	1176
20.72	1176
20.62	1177
20.62	1177
20.53	1177
20.44	1177
20.48	1177
20.39	1177
20.44	1177
20.58	1177
20.53	1177
20.53	1177
20.48	1177
20.58	1178
20.48	1178
20.48	1178
20.39	1178
20.34	1178
20.34	1178
20.29	1179
20.29	1179
20.29	1179
20.34	1179
20.44	1179
20.39	1179
20.44	1179



24.41	23.19	1400	429.00
24.41	23.19	1400	426.00
24.41	23.11	1400	433.00
24.41	23.27	1400	404.00
24.41	23.03	1400	387.00
24.41	23.27	1400	411.00
24.17	23.19	1400	377.00
24.17	23.19	1400	400.00
24.17	23.52	1400	378.00
24.17	23.11	1400	393.00
24.17	23.27	1400	387.00
24.17	23.11	1400	357.00
24.17	23.27	1400	392.00
24.17	23.27	1400	416.00
24.17	23.19	1400	437.00
24.17	23.19	1400	430.00
23.93	23.27	1400	438.00
24.17	23.27	1400	451.00
24.17	23.11	1400	450.00
24.17	23.11	1400	457.00
24.17	23.19	1400	460.00
24.17	23.19	1400	468.00
23.93	23.11	1400	470.00
23.93	23.27	1400	470.00
23.93	23.27	1400	471.00
23.93	23.27	1400	469.00
23.93	23.27	1400	468.00
23.93	23.27	1400	471.00
23.93	23.19	1400	471.00
23.93	23.27	1400	477.00
23.93	23.27	1400	458.00
23.93	23.11	1400	554.00
23.93	23.27	1400	542.00
23.93	23.19	1400	535.00
23.93	23.27	1400	566.00
23.93	23.52	1400	577.00

Pictures





The future of animal welfare

nimal welfare is quite a hot opic in this day and age, because animals aren't always treated well. We feel that it's necessary to look after animals n a more humane way. The igbit was our solution to this nimal welfare problem

e have chosen to measure the ealth and welbeing of the pigs by list of parameters. The most portant parameters that we want measure are: Movement of the igs, temperature of the pigs and he overall noise levels that are in a ig stable

Code

Our project consisted of the following codebases:

C# (Arduino): Hub, Node and Stable Sensor
Java (Processing): UI with classes

All code original from group members, or from libraries examples that were included.

Arduino Hub

```
//************************************************************
//
//    Forgotten Vegetables, HUB code
//
//************************************************************
//include libraries
#include "painlessMesh.h"

//define dependancies
#define   MESH_PREFIX     "testMesh"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555

//make objects
Scheduler userScheduler; // to control your personal task
painlessMesh  mesh;

// User stub
void sendMessage() ; // Prototype so PlatformIO doesn't complain

//network administration
void sendMessage(String msg) {
  mesh.sendBroadcast( msg );
}

// Needed for painless library
void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}

//network administration
```

```
void newConnectionCallback(uint32_t nodeId) {
  Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
}

//network administration
void changedConnectionCallback() {
  Serial.printf("Changed connections\n");
}

//network administration
void nodeTimeAdjustedCallback(int32_t offset) {
  Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}

void setup() {
  //open serial port
  Serial.begin(9600);

  //mesh.setDebugMsgTypes( ERROR | MESH_STATUS | CONNECTION | SYNC |
COMMUNICATION | GENERAL | MSG_TYPES | REMOTE ); // all types on
  mesh.setDebugMsgTypes( ERROR | STARTUP );  // set before init() so that you can see
startup messages

  //initialize mesh
  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
}

void loop() {
  // it will run the user scheduler as well
  mesh.update();
  serial();
}

//serial parser, passthrough to UI
void serial() {
    if( Serial.available() > 0) {
      String incomingSerial = Serial.readStringUntil('\n');
      sendMessage(incomingSerial);
      }
}
```

Arduino Node

```
//************************************************************
//
//    Forgotten Vegetables, NODE code
//
//************************************************************
//import libraries
#include "painlessMesh.h"
#include <FastLED.h>
#include<Wire.h>

//define dependancies
#define   MESH_PREFIX     "testMesh"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555

//make objects
Scheduler userScheduler; // to control your personal task
painlessMesh  mesh;
CRGB leds[1];

//define variables
const int MPU_addr = 0x68; // I2C address of the MPU-6050
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
int totalAcc;
int previousAcc;
int tolerance = 2000;
int steps = 15;
unsigned long previousMillis = 0;
const int wait = 100;

// User stub
void sendMessage() ; // Prototype so PlatformIO doesn't complain

// polling task
void sendMessage() {
  String msg = "I";
  msg += mesh.getNodeId();
  msg += "M";
  char* tempSteps;
  sprintf(tempSteps, "%04d", steps);
  msg += tempSteps;
  msg += "T";
  int tempTemp = Tmp  * 10;
  msg += tempTemp;
  mesh.sendBroadcast( msg );
}

// Needed for painless library
void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
  checkMessage(msg);
}

//protocol administration
void newConnectionCallback(uint32_t nodeId) {
  Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
}

//protocol administration
void changedConnectionCallback() {
  Serial.printf("Changed connections\n");
}

//protocol administration
void nodeTimeAdjustedCallback(int32_t offset) {
  Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}

void setup() {
  //open serial
  Serial.begin(9600);

  //mesh.setDebugMsgTypes( ERROR | MESH_STATUS | CONNECTION | SYNC |
COMMUNICATION | GENERAL | MSG_TYPES | REMOTE ); // all types on
  mesh.setDebugMsgTypes( ERROR | STARTUP );  // set before init() so that you can see
startup messages

  //initialize mesh network
  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
  //initialize LEDS
  FastLED.addLeds<WS2812, 0, GRB>(leds, 1);
  leds[0] = CRGB(255, 255, 255);
  FastLED.show();

  //initialize sensors
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
}

void loop() {
  // it will run the user scheduler as well
  mesh.update();
  //accelerometer
  accel;
}

//serial parser for incomign messages, changes LED color when not connected to node
void checkMessage(String incomingMesh) {
  if (incomingMesh.charAt(0) == 'P') {
    if (incomingMesh.charAt(1) == 'R') {
      leds[0] = CRGB(255, 0, 0);
      FastLED.show();
    } else if (incomingMesh.charAt(1) == 'G') {
      leds[0] = CRGB(0, 255, 0);
      FastLED.show();
    } else if (incomingMesh.charAt(1) == 'B') {
      leds[0] = CRGB(0, 0, 255);
      FastLED.show();
    } else {
      leds[0] = CRGB(255, 255, 255);
      FastLED.show();
    }

    sendMessage();
  }
}

//measures movement
```

```
void accel() {
  if (millis() > previousMillis + wait) { //timing
    previousMillis = millis();

    //start reading sensor data
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
    Wire.endTransmission(false);

    //read specific sensor data
    Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
    AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
    AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
    AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
    Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

    //calculate accelaration
    totalAcc = sqrt(AcX * AcX + AcY * AcY + AcZ * AcZ); //calculate total acceleration
    //Serial.print("Total accelleration: "); Serial.println(totalAcc);

    //check if step threshold is met
    if (totalAcc < previousAcc - tolerance) { //calculate if step was measured
      steps++;
      //Serial.print("Steps: "); Serial.println(steps);
    }

    //reset temporary variables
    previousAcc = totalAcc;
  }
}
```

Arduino Stable Sensor

```
//************************************************************
//
//          Forgotten Vegetables, STABLE
//
//************************************************************
//import libraries
#include "painlessMesh.h"
#include <SPI.h>
#include <MFRC522.h>

//define dependancies
#define   MESH_PREFIX     "whateverYouLike"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555

//define pins
#define sensorMain A0
#define analogDigPin1 16
#define analogDigPin2 5
#define analogDigPin3 4
#define RST_PIN 0
#define SS_PIN 2

//define variables
int noiseValue = 0;
float sensorValue1;
float sensorValue2;
float voltageOut;
float voltageOut2;
float temperatureC;
float temperatureC2;
float lightValue;
//float lightValue2;
unsigned long previousMillis;
String cardID;

//define objects
Scheduler userScheduler; // to control your personal task
painlessMesh  mesh;
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

```
// User stub
void sendMessage() ; // Prototype so PlatformIO doesn't complain
void polling();

//define tasks
Task taskSendMessage( TASK_SECOND * 1 , TASK_FOREVER, &sendMessage );
Task pollingTask( TASK_SECOND * 5 , TASK_FOREVER, &polling);

Task noiseTask((TASK_SECOND * 5) + 0.1 , TASK_FOREVER, &noise);
Task tempInTask((TASK_SECOND * 5) + 2.2 , TASK_FOREVER, &tempIn);
Task tempOutTask((TASK_SECOND * 5) + 4.3 , TASK_FOREVER, &tempOut);
Task lightTask((TASK_SECOND * 5) + 6.4 , TASK_FOREVER, &light);
//Task light2Task((TASK_SECOND * 5) + 8.5 , TASK_FOREVER, &light2);


//network administration
void sendMessage() {
  //  String msg = "Hello from node ";
  //  msg += mesh.getNodeId();
  //  mesh.sendBroadcast( msg );
}

// Needed for painless library
void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}

//network administration
void newConnectionCallback(uint32_t nodeId) {
  Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
}

//network administration
void changedConnectionCallback() {
  Serial.printf("Changed connections\n");
}

//network administration
void nodeTimeAdjustedCallback(int32_t offset) {
  Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}
```

```
void setup() {
  //Open serial
  Serial.begin(115200);
  SPI.begin();

  //initialize pins
  pinMode(analogDigPin1, OUTPUT);
  pinMode(analogDigPin2, OUTPUT);
  pinMode(analogDigPin3, OUTPUT);
  pinMode(sensorMain, INPUT);

  //initialize RFID
  mfrc522.PCD_Init();

  //mesh.setDebugMsgTypes( ERROR | MESH_STATUS | CONNECTION | SYNC | COMMUNICATION | GENERAL | MSG_TYPES | REMOTE ); // all types on
  mesh.setDebugMsgTypes( ERROR | STARTUP );  // set before init() so that you can see startup messages

  //initialize mesh network
  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);

  //define tasks
  userScheduler.addTask( taskSendMessage );
  taskSendMessage.enable();

  userScheduler.addTask( pollingTask );
  pollingTask.enable();


  userScheduler.addTask( noiseTask );
  noiseTask.enable();
  userScheduler.addTask( tempInTask );
  tempInTask.enable();
  userScheduler.addTask( tempOutTask );
  tempOutTask.enable();
  userScheduler.addTask( lightTask );
  lightTask.enable();
  //userScheduler.addTask( light2Task );
```

```
 // light2Task.enable();
}

void loop() {
  // it will run the user scheduler as well
  mesh.update();
  //update RFID
  rfid();
}

//RFID reader
void rfid() {
  if (millis() > previousMillis + 100) { //timing with millis() instead of delay()
    previousMillis = millis();

    //reset cardID string
    cardID = "";
    MFRC522::StatusCode status;

    // Reset the loop if no new card present on the sensor/reader. This saves the entire process
when idle.
    if ( ! mfrc522.PICC_IsNewCardPresent()) {
      return;
    }

    // Select one of the cards
    if ( ! mfrc522.PICC_ReadCardSerial()) {
      return;
    }

    //read first 4 bytes (which contain UID of card) and convert to String
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      cardID += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
      cardID += String(mfrc522.uid.uidByte[i], HEX);
    }

    //if RFID is found, send message to network
    String msg = "R";
    msg += cardID;
    Serial.println(msg);
    mesh.sendBroadcast(msg);

    //stop reading
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();
  }
}

//timed polling
void polling() {
  //verstuur data
  String msg;
  msg += "S"; //stable
  msg += "I";  //temperature
  msg += temperatureC; // 3 values
  msg += "O";
  msg += temperatureC2; //3 values
  msg += "N"; // noise level
  msg += noiseValue;
  msg += "L"; // Light intensity
  msg += lightValue;
 /* msg += "D"; //Light outside ("met de D van darkness" - Hilke 2020)
  msg += lightValue2; */

  Serial.println(msg);
  mesh.sendBroadcast(msg);
}

//temperature measurement 1
void tempIn() {
  digitalWrite(16, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
  sensorValue1 = analogRead(sensorMain);
  voltageOut = (sensorValue1 * 5000) / 1024;

  // calculate temperature for LM35 (LM35DZ)
  temperatureC = voltageOut / 15;
  Serial.print("tempIn = "); Serial.println(temperatureC);
}

//temperature measurement 2
void tempOut() {
  digitalWrite(16, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
  sensorValue2 = analogRead(sensorMain);
  voltageOut2 = (sensorValue2 * 5000) / 1024;

  // calculate temperature for LM35 (LM35DZ)
  temperatureC2 = voltageOut2 / 60;
  Serial.print("tempOut = "); Serial.println( temperatureC2);
}

//noise measurement
void noise() {
  digitalWrite(16, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
  noiseValue = analogRead (sensorMain);
  Serial.print("Noise = "); Serial.println (noiseValue / 8, DEC);
}

//light measurement
void light() {
  digitalWrite(16, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, HIGH);
  lightValue = analogRead(sensorMain);

  Serial.print("Light1 = ");
  Serial.print(lightValue);   // the raw analog reading

  // We'll have a few threshholds, qualitatively determined
  if (lightValue < 10) {
    Serial.println(" - Dark");
  } else if (lightValue < 200) {
    Serial.println(" - Dim");
  } else if (lightValue < 500) {
    Serial.println(" - Light");
  } else if (lightValue < 800) {
    Serial.println(" - Bright");
  } else {
    Serial.println(" - Very bright");
  }
}
```

Processing UI

```
/*This code represents the screen which the user gets to see. It contains all obtained
information
 that we measure in the stable and from the pigs. Screen one contains rectangles which each
represent
 one of the pigs. When clicked on one of the rectangles, you get a second screen which shows
more information
 about the pig.*/

import processing.serial.*;
Serial port;

Table stable;
int screen = 1;

UI_1 firstui;
UI_2 secui;
UI_3 thirdui;
Stable stableobject;

UI_manager uimanager;
Serial_manager serial_manager;

//Text specifics and the text that the pig is healthy or unhealthy
PFont f;
String nothealthy = "Not Healthy";
String healthy = "Healthy";

boolean welfare1 = true;
boolean welfare2 = true;
boolean welfare3 = true;
float temp;
float amountsteps;

float xstable;
float x;
float x1;
float x2;
float x3;

PImage backarrow;
```

```
float arrowX;
float arrowY;

void setup() {
  //size and backgroundcolor
  size(1400, 900);
  background(220);

  //Font specifications
  f = createFont("Calibri", 70);
  fill(255);
  textFont(f, 70);

  firstui = new UI_1(loadImage("pig.png"), loadImage("backarrow.png"));
  secui = new UI_2(loadImage("pig.png"), loadImage("backarrow.png"));
  thirdui = new UI_3(loadImage("pig.png"), loadImage("backarrow.png"));
  stableobject = new Stable(loadImage("pig.png"), loadImage("stable.png"),
loadImage("backarrow.png"));
  backarrow = loadImage("backarrow.png");

  //port = new Serial(this, Serial.list()[0], 9600);
  uimanager = new UI_manager(loadImage("backarrow.png"));
  serial_manager = new Serial_manager();
  port = new Serial(this, Serial.list()[0], 9600);
  println("initialized.");

  //load csv file of the stable
  stable = loadTable("stable.csv", "header");

  arrowX = 50;
  arrowY = 50;
}

void draw() {
  serial_manager.display();

  //For loading the table of the stable and conluding the x already
  //The text won't be displayed when the screen is not 3
  stableobject.updateX();
  //shows first screen with the option of pigs or stable
  if (screen == 1) {
    stableobject.display();
  }
  x1 = x + x1;
  x2 = x + x2;
  x3 = x + x3;

  if (x1 >= 2) {
    welfare1 = false;
  }
  if (x2 >= 2) {
    welfare2 = false;
  }
  if (x3 >= 2) {
    welfare3 = false;
  }
}

void mousePressed() {

  if (screen == 1) {
    //clicks on pig option
    if (mouseX >= 150 && mouseX <= 650 && mouseY >= 100 && mouseY <= 800) {
      screen += 1;

      //clicks on stable option
    } else if (mouseX >= 750 && mouseX <= 1250 && mouseY >= 100 && mouseY <= 800) {
      screen += 2;
    }
  }

  // when the first choise is made the following happens
  if (screen == 2) {
    background(200);
    uimanager.display();

    stroke(255);
    fill(255);
    ellipse(50, 50, 50, 50);
    imageMode(CENTER);
    image(backarrow, 50, 50, 40, 40);

    //When you click on the arrow to go back, the pigs are all shown again
    if (mouseX >= 10 && mouseX <= 90 && mouseY >= 10 && mouseY <= 90) {
      background(220);
      screen = 1;
    }
    //if clicked on one of the squares, specifics of that specific pig are shown
    if (mouseX >= 20 && mouseX <= 350) {
      //first ui (pig)
      if (mouseY >= 120 && mouseY <= 290) {
        screen = 4;
        firstui.specifics();

        //second ui
      } else if (mouseY >= 310 && mouseY <= 480) {
        screen = 5;
        secui.specifics();

        //third ui
      } else if (mouseY >= 498 && mouseY <= 668) {
        screen = 6;
        thirdui.specifics();
      }
      }
  }

  //If chosen the stable option the measurements are shown.
  if (screen == 3) {
    stableobject.specifics();
    stableobject.updateX();

    //The back arrow in the upperleft corner
    stroke(255);
    fill(255);
    ellipse(arrowX, arrowY, 50, 50);
    imageMode(CENTER);
    image(backarrow, arrowX, arrowY, 40, 40);


    if (mouseX >= 10 && mouseX <= 90 && mouseY >= 10 && mouseY <= 90) {
      background(220);
      screen = 1;
    }
  }

  if (screen == 4 | screen == 5 | screen == 6) {
    //The back arrow in the upperleft corner
```

```
    stroke(255);
    fill(255);
    ellipse(arrowX, arrowY, 50, 50);
    imageMode(CENTER);
    image(backarrow, arrowX, arrowY, 40, 40);


    if (mouseX >= 10 && mouseX <= 90 && mouseY >= 10 && mouseY <= 90) {
      screen = 2;
    }
    }
}
```

Serial manager

```
class Serial_manager {
  String nodeID;
  String temp;
  String steps;

  String tempIn;
  String tempOut;
  String noise;
  String light;

  Serial_manager() {
    println("Available serial ports:");
    for (int i = 0; i<Serial.list().length; i++) {
      print("[" + i + "] ");
      println(Serial.list()[i]);
    }
  }

  void display() {
    while (port.available () > 0) {
      String msg = trim(port.readStringUntil('\n'));
      if (msg == null) {
        print("error - MESSAGE EMPTY ");
      } else if (msg.length() < 1) {
        println("error 0string");
      } else if (msg.charAt(0) == 'I') {

        println("incoming from node: " + msg);

        if (msg.charAt(11) == 'T') {
          nodeID = msg.substring(1, 11);
          temp = msg.substring(12, 17);
          steps = msg.substring(18);
        } else if (msg.charAt(10) == 'T') {
          nodeID = msg.substring(1, 10);
          temp = msg.substring(11, 16);
          steps = msg.substring(17);
        }

        String filename = nodeID + ".csv";
        String timeNow = day() + "" + month() + "" + year() + "" + hour() + "" + minute() + "" +
second();

        Table table = loadTable(filename, "header");
        TableRow newRow = table.addRow();
        newRow.setString("time", timeNow);
        newRow.setString("temp", temp);
        newRow.setString("steps", steps);
        saveTable(table, filename);
      } else if (msg.charAt(0) == 'R') {

        println("incoming from rfid: " + msg);
        msg = msg.substring(1);
        println("RFID tag: " + msg);

        Table table = loadTable("idList.csv", "header");
        boolean found = false;
        //for (TableRow row : table.rows()) {
        for (int row = 0; row < table.getRowCount(); row++) {
          //println(row);
          if (table.getString(row, "RFID").equals(msg)) {

            String location = table.getString(row, "location");
            println("RFID found: " + msg + " location: " + location);

            if (location.equals("inside")) {
              table.setString(row, "location", "outside");
            } else {
              table.setString(row, "location", "inside");
            }

            saveTable(table, "idList.csv");
            found = true;
           }
        }
        if (!found) {
          println("error - RFID not found");
         }
      } else if (msg.charAt(0) == 'S') {

        println("incoming from stable: " + msg);

        tempIn = msg.substring(msg.indexOf("SI")+2, msg.indexOf("O"));
        tempOut = msg.substring(msg.indexOf("O")+1, msg.indexOf("N"));
        noise = msg.substring(msg.indexOf("N")+1, msg.indexOf("L"));
        light = msg.substring(msg.indexOf("L")+1);
        String timeNow = day() + "" + month() + "" + year() + "" + hour() + "" + minute() + "" +
second();

        Table table = loadTable("stalSensor.csv", "header");
        TableRow newRow = table.addRow();
        newRow.setString("time", timeNow);
        newRow.setString("tempIn", tempIn);
        newRow.setString("tempOut", tempOut);
        newRow.setString("noise", noise);
        newRow.setString("light", light);
        saveTable(table, "stalSensor.csv");
      } else {
        println("error - INVALID MESSAGE: " + msg);
       }
    }
  }
}
```

Stable

```
class Stable {

  float prevX = 200;
  float prevY = 0;
  float avg = 0;
  float prevX2 = 200;
  float prevY2 = 0;
  float avg2 = 0;
  float emptyRows2 = 0;


  float intemp = 0;
  float outtemp = 0;
  float inlight = 0;
  float sound = 0;
  float emptyRows = 0;
  float avtemp = 0;


  //The images which are used for the first screen option
  PImage pig;
  PImage stablepic;

  //These are for the image coördinates
  float pigX;
  float pigY;
  float stableX;
  float stableY;

  float stableTemp;
  float noise;
  float light;


  //Text for the option to choose in the first screen
  String choosepig = "Pigs";
  String choosestable = "Stable";

  Stable(PImage _pig, PImage _stable, PImage _backarrow) {
    backarrow = _backarrow;
    stablepic = _stable;
    pig = _pig;
    pigX = 400;
    pigY = 350;
    stableX = 1002;
    stableY = 320;
  }


  void display() {

    //First screen
    //The two black big rectangles
    fill(0);
    stroke(0);
    rect(150, 100, 500, 700);
    rect(750, 100, 500, 700);

    //The two smaller white rectangles behind the pictures
    stroke(255);
    fill(255);
    rect(200, 150, 400, 350);
    rect(800, 150, 400, 350);

    //The images which are places in the white rectangles
    imageMode(CENTER);
    image(pig, pigX, pigY, 300, 300);
    image(stablepic, stableX, stableY, 400, 400);

    //The text, one which says 'stable' and the other 'pigs'
    textSize(100);
    fill(255);
    textAlign(CENTER);
    text(choosepig, 390, 600);
    text(choosestable, 1000, 600);

    textSize(80);
    fill(0);
    textAlign(CENTER);
    text("PigBit", width/2, 70);
  }

  void specifics() {
    //When the optionstable turns true this void is activated
    background(0);
    stroke(255);
    fill(255);
    rect(200, 100, 1000, 800);

    //Text for the information about the stable
    fill(0);
    textSize(50);
    textAlign(CENTER);
    text("Information about Stable", 805, 200);

    //The rectangle behind the image of the stable
    fill(255);
    stroke(0);
    rect(220, 165, 305, 265);

    //The image display of the stable
    imageMode(CENTER);
    image(stablepic, 370, 300, 300, 300);

    //specific info pigs
    fill(0);
    textAlign(LEFT);
    textSize(25);
  }

  void updateX() {
    fill(0);
    if (stable.getRowCount() >= 100) {
      if (screen == 3) {
        grapher();
       }
    } else {
      println("too little data to graph");
    }

    //The temperature inside the stable
    TableRow rowNr = stable.getRow(stable.getRowCount() - 1);
    intemp = rowNr.getFloat("tempIn");

    if (screen == 3) {
```

```
  text("Indoor temperature: " + intemp, 550, 300);
}

if (intemp < 17 && intemp > 23) {
  stableTemp = 1;
} else {
  stableTemp = 0;
}

//reset values
intemp = 0;
emptyRows = 0;

//Adding all the row information for the inside temperature
for (TableRow row : stable.rows()) {
  if (row.getFloat("tempIn") == row.getFloat("tempIn")) {
    avtemp += row.getFloat("tempIn");
  } else {
    emptyRows++;
   }
}

//Taking the average of all the temperature values together
avtemp = avtemp / (stable.getRowCount() - emptyRows);

if (screen == 3) {
  text("Average indoor temperature: " + avtemp, 220, 500);
}

avtemp = 0;
emptyRows = 0;


//The temperature outside of the stable
rowNr = stable.getRow(stable.getRowCount() - 1);
outtemp = rowNr.getFloat("tempOut");

if (screen == 3) {
  text("Outdoor temperature: " + outtemp, 550, 340);
}

outtemp = 0;
emptyRows = 0;
```

```
//Adding all the row information for the inside temperature
for (TableRow row : stable.rows()) {
  if (row.getFloat("tempOut") == row.getFloat("tempOut")) {
    avtemp += row.getFloat("tempOut");
  } else {
    emptyRows++;
    }
}

//Taking the average of all the temperature values together
avtemp = avtemp / (stable.getRowCount() - emptyRows);

if (screen == 3) {
  text("Average outside temperature: " + avtemp, 220, 540);
}

avtemp = 0;
emptyRows = 0;

//The light inside the stable
rowNr = stable.getRow(stable.getRowCount() - 1);
inlight = rowNr.getFloat("light");

if (screen == 3) {
  text("The amount of light in the stable: " + inlight, 550, 380);
}

if (inlight > 200 && inlight < 800) {
  light = 0;
} else {
  light = 1;
}

inlight = 0;
emptyRows = 0;

//Adding all the row information for the inside temperature
for (TableRow row : stable.rows()) {
  if (row.getFloat("light") == row.getFloat("light")) {
    inlight += row.getFloat("light");
  } else {
    emptyRows++;
    }
}

//Taking the average of all the temperature values together
inlight = inlight / (stable.getRowCount() - emptyRows);

if (screen == 3) {
  text("Average light level inside: " + inlight, 220, 580);
}

inlight = 0;
emptyRows = 0;


//The sound inside the stable
rowNr = stable.getRow(stable.getRowCount() - 1);
sound = rowNr.getFloat("noise");

if (screen == 3) {
  text("Soundlevel: " + sound, 550, 420);
}

if (sound > 85) {
  noise = 1;
} else {
  noise = 0;
}

sound = 0;
emptyRows = 0;

//Adding all the row information for the inside temperature
for (TableRow row : stable.rows()) {
  if (row.getFloat("noise") == row.getFloat("noise")) {
    sound += row.getFloat("noise");
  } else {
    emptyRows++;
    }
}

//Taking the average of all the temperature values together
sound = sound / (stable.getRowCount() - emptyRows);
```

```
  if (screen == 3) {
    text("Average noise level: " + sound, 220, 620);
  }

  sound = 0;
  emptyRows = 0;


  x = light + noise + stableTemp;
}

void grapher() {
  // Shows the graphs on the bottom of the screen with text
  for (int row = stable.getRowCount()-100; row < stable.getRowCount(); row++) {
    if (stable.getFloat(row, "tempIn") == stable.getFloat(row, "tempIn")) {
      avg += stable.getFloat(row, "tempIn");
    } else {
      emptyRows++;
    }
    if (stable.getFloat(row, "tempOut") == stable.getFloat(row, "tempOut")) {
      avg2 += stable.getFloat(row, "tempOut");
    } else {
      emptyRows2++;
      }
  }

  avg = avg / (100 - emptyRows);
  println(avg);
  avg2 = avg2 / (100 - emptyRows2);
  println(avg);

  for (int row = stable.getRowCount()-100; row < stable.getRowCount(); row++) {
    stroke(165, 32, 25);
    float curY = map(stable.getFloat(row, "tempIn"), avg-1, avg+5, 900, 890);

    if (prevY == 0) {
      prevY = curY;
    }

    line(prevX, prevY, prevX+10, curY);
    prevX += 10;
    prevY = curY;
```

```
      stroke(34, 139, 34);
      float curY2 = map(stable.getFloat(row, "tempOut"), avg2-7, avg2+1, 900, 890);

      if (prevY2 == 0) {
        prevY2 = curY2;
      }

      line(prevX2, prevY2, prevX2+14, curY2);
      prevX2 += 14;
      prevY2 = curY2;

    // text and graph lines for the graph
      stroke(0);
      line(width-205, 800, width-205, 897);
      line(10, 897, width-205, 897);
      fill(0);
      text("stable.csv" + " temperature graph", 700, 700);
      fill(165, 32, 25);
      text("red temperature graph =" + " Inside temperature", 700, 725);
      fill(34, 139, 34);
      text("green temperature graph =" + " Outside temperature", 700, 750);
      fill(0);
      text("30C", 1125, 800);
      text("10C", 1125, 894);
      fill(0);
      rect(1200, 0, 1200, 1000);

      avg = 0;
      emptyRows = 0;
      avg2 = 0;
      emptyRows2 = 0;
    }
  }
}
```

UI_1

```
/*This is the first UI that is displayed. Every UI belongs to one pig and
 contains the information about this pig.*/

class UI_1 {
  Table table1;

  float prevX = 200;
  float prevY = 0;
  float avg = 0;

  float avtemp = 0;
  float emptyRows = 0;
  float steps;

  String pigid = "pig1";
  String fileName = pigid + ".csv";

  int w = 350;
  int h = 170;
  int l = 20;
  int m = 40;
  int r = 100;
  float s = 100;

  //Image of the pig
  PImage pig;

  //These are used for the coördinates of the images
  float pigX;
  float pigY;

  UI_1(PImage _pig, PImage _backarrow) {
    pig = _pig;
    backarrow = _backarrow;
    pigX = 80;
    pigY = 180;
  }

  void display() {
    //shows the rectangle of the first pig
    if (screen == 2) {
      //Black rectangle
      fill(0);
      stroke(0);
      rect(l, m + 80, w, h);

      //White circle for behind the pig picture
      stroke(255);
      fill(255);
      ellipse(80, s + 80, r, r);

      //little white rectangle with info
      fill(255);
      rect(150, 1.2*m + 80, 210, 150);

      //The picture of the pig
      imageMode(CENTER);
      image(pig, pigX, pigY, 70, 75);

      //text in white rectangle
      textSize(50);
      if (welfare1 == true) {
        fill(78, 204, 53);
        text(healthy, 250, 180);
      } else {
        textSize(40);
        fill(227, 39, 39);
        text(nothealthy, 250, 180);
        }
    }
  }

  void specifics() {
    //load the table with information about pig1
    table1 = loadTable(fileName, "header");

    //If the user clicked on one of the rectangles of a pig, the specific info about this pig is shown
as followed

    //The white and black 'page' background
    background(0);
    stroke(255);
    fill(255);
```

```
rect(200, 100, 1000, 800);

//The picture of the pig
stroke(0);
rect(260, 150, 220, 220);
imageMode(CENTER);
image(pig, 370, 270, 200, 200);

fill(255);
ellipse(50, 50, 50, 50);
imageMode(CENTER);
image(backarrow, arrowX, arrowY, 40, 40);

//When you click on the arrow to go back, the pigs are all shown again
if (mouseX >= 10 && mouseX <=90 && mouseY >= 10 && mouseY <= 90) {
  background(220);
  screen = 2;
}

//The text
fill(0);
textAlign(CENTER);
textSize(50);
text("Information about " + pigid, 755, 200);

//specific info pigs
textAlign(LEFT);
textSize(25);

//Adding all the row information
for (TableRow row : table1.rows()) {
  if (row.getFloat("temp") == row.getFloat("temp")) {
    avtemp += row.getFloat("temp");
  } else {
    emptyRows++;
    }
}

//Taking the average of all the temperature values together
avtemp = avtemp / (table1.getRowCount() - emptyRows);
text("Average temperature: " + avtemp, 520, 280);

avtemp = 0;
```

```
emptyRows = 0;


//The actual temperature of the pig
TableRow rowNr = table1.getRow(table1.getRowCount() - 1);
avtemp = rowNr.getFloat("temp");

text("Actual temperature: " + avtemp, 520, 320);

if (avtemp > 39 && avtemp < 37) {
  temp = 1;
} else {
  temp = 0;
}


avtemp = 0;
emptyRows = 0;


//The total amount of steps done by one pig
rowNr = table1.getRow(table1.getRowCount() - 1);
steps = rowNr.getFloat("steps");

text("Total steps: " + steps, 520, 360);

if (steps < 200) {
  amountsteps = 1;
} else {
  amountsteps = 0;
}

steps = 0;
emptyRows = 0;

x1 = amountsteps + temp;

text("Status Pig = ", 250, 500);
if (welfare1 == true) {
  textSize(60);
  fill(78, 204, 53);
  text(healthy, 380, 500);
} else {
      textSize(60);
      fill(227, 39, 39);
      text(nothealthy, 380, 500);
    }

    if (table1.getRowCount() >= 100) {
      if (screen == 4) {
        grapher();
      }
    } else {
      println("too little data to graph");
    }
  }

  void grapher() {
    // Shows the graph on the bottom of the screen with text
    for (int row = table1.getRowCount()-100; row < table1.getRowCount(); row++) {
      if (table1.getFloat(row, "temp") == table1.getFloat(row, "temp")) {
        avg += table1.getFloat(row, "temp");
      } else {
        emptyRows++;
        }
    }

    avg = avg / (100 - emptyRows);
    println(avg);

    for (int row = table1.getRowCount()-100; row < table1.getRowCount(); row++) {
      stroke(165, 32, 25);
      float curY = map(table1.getFloat(row, "temp"), avg-1, avg+5, 900, 890);

      if (prevY == 0) {
        prevY = curY;
      }

      line(prevX, prevY, prevX+10, curY);
      prevX += 10;
      prevY = curY;

// text and graph lines for the graph
      stroke(0);
      line(width-205, 800, width-205, 897);
      line(10, 897, width-205, 897);
      textSize(20);
      fill(0);
      text("table1.csv" + " temperature graph", 700, 700);
      fill(0);
      textSize(12);
      text("30C", 1125, 800);
      text("10C", 1125, 894);
      fill(0);
      rect(1200, 0, 1200, 1000);

      avg = 0;
      emptyRows = 0;
    }
  }
}
```

UI_2

```
/*This is the first UI that is displayed. Every UI belongs to one pig and
 contains the information about this pig.*/

class UI_2 {
  Table table2;
  float avtemp = 0;
  float emptyRows = 0;
  float steps;

  float prevX = 200;
  float prevY = 0;
  float avg = 0;


  int w = 350;
  int h = 170;
  int l = 20;
  int m = 40;
  int r = 100;
  float s = 100;

  String pigid = "pig2";
  String fileName = pigid + ".csv";

  //Image of the pig
  PImage pig;

  //These are used for the coördinates of the images
  float pigX;
  float pigY;

  UI_2(PImage _pig, PImage _backarrow) {
    pig = _pig;
    backarrow = _backarrow;
    pigX = 80;
    pigY = 365;
  }

  void display() {
    //shows the rectangle of the second pig
    //Black rectangle
    fill(0);
    stroke(0);
    rect(l, 5.6*m + 85, w, h);

    //White circle for behind the pig picture
    stroke(255);
    fill(255);
    ellipse(80, 2.8*s + 85, r, r);

    //little white rectangle with info
    fill(255);
    rect(150, 5.8*m + 85, 210, 150);

    //The picture of the pig
    imageMode(CENTER);
    image(pig, pigX, pigY, 70, 75);

    //text in white rectangle
    if (welfare2 == true) {
      textSize(50);
      fill(78, 204, 53);
      text(healthy, 250, 365);
    } else {
      textSize(40);
      fill(227, 39, 39);
      text(nothealthy, 250, 365);
    }
  }

  void specifics() {
    //load the table with information about pig2
    table2 = loadTable(fileName, "header");
    //If the user clicked on one of the rectangles of a pig, the specific info about this pig is shown as followed
    if (screen == 5) {

      background(0);
      stroke(255);
      fill(255);
      rect(200, 100, 1000, 800);
```

```
//The picture of the pig
stroke(0);
rect(260, 150, 220, 220);
imageMode(CENTER);
image(pig, 370, 270, 200, 200);

//The text
fill(0);
textAlign(CENTER);
textSize(50);
text("Information about " + pigid, 755, 200);

//specific info pigs
textAlign(LEFT);
textSize(25);

for (TableRow row : table2.rows()) {
  if (row.getFloat("temp") == row.getFloat("temp")) {
    avtemp += row.getFloat("temp");
  } else {
    emptyRows++;
    }
}

//get average temperature
avtemp = avtemp / (table2.getRowCount() - emptyRows);

//display text
text("Average temperature: " + avtemp, 520, 280);

//clear values
avtemp = 0;
emptyRows = 0;

//The actual temperature of the pig
TableRow rowNr = table2.getRow(table2.getRowCount() - 1);
avtemp = rowNr.getFloat("temp");

//display text
text("Actual temperature: " + avtemp, 520, 320);

if (avtemp > 39 && avtemp < 37) {
  temp = 1;
```

```
} else {
  temp = 0;
}

//clear values
avtemp = 0;
emptyRows = 0;

//The total amount of steps done by one pig
rowNr = table2.getRow(table2.getRowCount() - 1);
steps = rowNr.getFloat("steps");

text("Total steps: " + steps, 520, 360);

if (steps < 200) {
  amountsteps = 1;
} else {
  amountsteps = 0;
}

steps = 0;
emptyRows = 0;

x2 = amountsteps + temp;

//show if pig is healthy or not
text("Status Pig = ", 250, 500);
if (welfare2 == true) {
  textSize(60);
  fill(78, 204, 53);
  text(healthy, 380, 500);
} else {
  textSize(60);
  fill(227, 39, 39);
  text(nothealthy, 380, 500);
}

if (table2.getRowCount() >= 100) {
  if (screen == 5) {
    grapher();
    }
} else {
  println("too little data to graph");
```

```
      }
  }
}

void grapher() {
  // Shows the graph on the bottom of the screen with text
  for (int row = table2.getRowCount()-100; row < table2.getRowCount(); row++) {
    if (table2.getFloat(row, "temp") == table2.getFloat(row, "temp")) {
      avg += table2.getFloat(row, "temp");
    } else {
      emptyRows++;
      }
  }

  avg = avg / (100 - emptyRows);
  println(avg);

  for (int row = table2.getRowCount()-100; row < table2.getRowCount(); row++) {
    stroke(165, 32, 25);
    float curY = map(table2.getFloat(row, "temp"), avg-1, avg+5, 900, 890);

    if (prevY == 0) {
      prevY = curY;
    }

    line(prevX, prevY, prevX+10, curY);
    prevX += 10;
    prevY = curY;

  // text and graph lines for the graph
    stroke(0);
    line(width-205, 800, width-205, 897);
    line(10, 897, width-205, 897);
    textSize(20);
    fill(0);
    text("table2.csv" + " temperature graph", 700, 700);
    fill(0);
    textSize(12);
    text("30C", 1125, 800);
    text("10C", 1125, 894);
    fill(0);
    rect(1200, 0, 1200, 1000);
```

```
      avg = 0;
      emptyRows = 0;
    }
  }
}
```

UI_3

```
/*This is the first UI that is displayed. Every UI belongs to one pig and
 contains the information about this pig.*/

class UI_3 {
  Table table3;
  float avtemp = 0;
  float emptyRows = 0;
  float steps;

  float prevX = 200;
  float prevY = 0;
  float avg = 0;

  int w = 350;
  int h = 170;
  int l = 20;
  int m = 40;
  int r = 100;
  float s = 100;

  String pigid = "pig3";
  String fileName = pigid + ".csv";

  //Image of the pig
  PImage pig;

  //These are used for the coördinates of the images
  float pigX;
  float pigY;

  UI_3(PImage _pig, PImage _backarrow) {
    pig = _pig;
    backarrow = _backarrow;
    pigX = 80;
    pigY = 555;
  }

  void display() {
    //shows the rectangle of the third pig
    //Black rectangle
    fill(0);
    stroke(0);
    rect(l, 10.2*m + 90, w, h);

    //White circle for behind the pig picture
    stroke(255);
    fill(255);
    ellipse(80, 4.65*s + 90, r, r);

    //little white rectangle with info
    fill(255);
    rect(150, 10.45*m + 90, 210, 150);

    //The picture of the pig
    imageMode(CENTER);
    image(pig, pigX, pigY, 70, 75);

    //text in white rectangle
    if (welfare3 == true) {
      textSize(50);
      fill(78, 204, 53);
      text(healthy, 250, 560);
    } else {
      textSize(40);
      fill(227, 39, 39);
      text(nothealthy, 250, 560);
    }
  }


  void specifics() {
    //load the table with information about pig3
    table3 = loadTable(fileName, "header");
    //If the user clicked on one of the rectangles of a pig, the specific info about this pig is shown
as followed
    if (screen == 6) {
      background(0);
      stroke(255);
      fill(255);
      rect(200, 100, 1000, 800);

      //The picture of the pig
      stroke(0);
      rect(260, 150, 220, 220);
      imageMode(CENTER);
      image(pig, 370, 270, 200, 200);
    }

    //The text
    fill(0);
    textAlign(CENTER);
    textSize(50);
    text("Information about " + pigid, 755, 200);

    //specific info pigs
    textAlign(LEFT);
    textSize(25);

    //Add all temperature values
    for (TableRow row : table3.rows()) {
      if (row.getFloat("temp") == row.getFloat("temp")) {
        avtemp += row.getFloat("temp");
      } else {
        emptyRows++;
      }
    }

    //Get the average temperature
    avtemp = avtemp / (table3.getRowCount() - emptyRows);

    text("Average temperature: " + avtemp, 520, 280);

    if (avtemp > 39 && avtemp < 37) {
      temp = 1;
    } else {
      temp = 0;
    }

    avtemp = 0;
    emptyRows = 0;


    //The actual temperature of the pig
    TableRow rowNr = table3.getRow(table3.getRowCount() - 1);
    avtemp = rowNr.getFloat("temp");
```

```
  text("Actual temperature: " + avtemp, 520, 320);

  avtemp = 0;
  emptyRows = 0;

  //The total amount of steps done by one pig
  rowNr = table3.getRow(table3.getRowCount() - 1);
  steps = rowNr.getFloat("steps");

  text("Total steps: " + steps, 520, 360);

  if (steps < 200) {
    amountsteps = 1;
  } else {
    amountsteps = 0;
  }

  steps = 0;
  emptyRows = 0;

  x3 = amountsteps + temp;

  //Display health status pig3
  text("Status Pig = ", 250, 500);
  if (welfare3 == true) {
    textSize(60);
    fill(78, 204, 53);
    text(healthy, 380, 500);
  } else {
    textSize(60);
    fill(227, 39, 39);
    text(nothealthy, 380, 500);
  }

  //Graph display
  if (table3.getRowCount() >= 100) {
    if (screen == 6) {
      grapher();
    }
  } else {
    println("too little data to graph");
  }
}
```

```
void grapher() {
  // Shows the graph on the bottom of the screen with text
  for (int row = table3.getRowCount()-100; row < table3.getRowCount(); row++) {
    if (table3.getFloat(row, "temp") == table3.getFloat(row, "temp")) {
      avg += table3.getFloat(row, "temp");
    } else {
      emptyRows++;
    }
  }

  avg = avg / (100 - emptyRows);
  println(avg);

  for (int row = table3.getRowCount()-100; row < table3.getRowCount(); row++) {
    stroke(165, 32, 25);
    float curY = map(table3.getFloat(row, "temp"), avg-1, avg+5, 900, 890);

    if (prevY == 0) {
      prevY = curY;
    }

    line(prevX, prevY, prevX+10, curY);
    prevX += 10;
    prevY = curY;

    // text and graph lines for the graph
    stroke(0);
    line(width-205, 800, width-205, 897);
    line(10, 897, width-205, 897);
    textSize(20);
    fill(0);
    text("table3.csv" + " temperature graph", 700, 700);
    fill(0);
    textSize(12);
    text("30C", 1125, 800);
    text("10C", 1125, 894);
    fill(0);
    rect(1200, 0, 1200, 1000);

    avg = 0;
    emptyRows = 0;
  }
```

```
 }
}
```

UI_manager

```
/*This class displays every UI on the screen*/

class UI_manager {
  UI_manager(PImage _backarrow) {
    backarrow = _backarrow;
  }

  void display() {

    //Black menu balk
    fill(0);
    stroke(0);
    rect(0, 0, 1800, 85);

    //The back arrow in the upperleft corner
    fill(255);
    ellipse(50, 50, 50, 50);
    imageMode(CENTER);
    image(backarrow, arrowX, arrowY, 50, 50);

    //display of all the UI's
    firstui.display();
    secui.display();
    thirdui.display();

    stroke(0);
    fill(0);
    rect(0, 800, 1800, 900);
  }
}
```