Proof of concept:

An Umbrella as a Mobile Acoustic Rain Gauge for use in Urban Areas

G.H. Gerritsen
G.H.Gerritsen@student.tudelft.nl, 4274164
Supervisors: dr. ir. R.W. Hut, dr. ir. D. van Halem

Water Resources Section, Faculty of Civil Engineering and Geosciences,
Delft University of Technology, Delft, the Netherlands

April 2020

Abstract. *To validate rainfall intensity in urban areas measured by satellite a first prototype proof of concept is introduced and tested. Using low-cost electronics an umbrella is converted into a mobile acoustic rain gauge which can be used in urban areas to measure rainfall intensities. A reed switch is placed in the umbrella to measure whether the umbrella is open or closed. Using a piezoelectric sensor and a Sparkfun sound detector rain droplets are detected and using a Pycom WiPy send over Bluetooth to an application which saves it on an online server. Tests during a laboratory experiment to see how the output data evolves shows that the data collected have an output range of about 10 % compared to its mean value. During field evaluation, to compare its output data with rainfall intensities as measured with radar, it is shown that the output data follows the radar measurements within acceptable bounds.*

1. Introduction

To measure rainfall intensities, three main options are available: rain gauges, radars and satellite data. With radar and satellite data the rainfall over a large area can be determined. However, as the grid size of these methods are from hundreds of meters up to several kilometres the results of these measurements are in many cases not accurate enough. For a more accurate analysis these results should be validated with ground measurements collected by rain gauges. These rain gauges provide rainfall measurements at ground level at a specific point or very small area. Rain gauge stations must meet specific international norms to be considered as an official recording rain gauge (KNMI, 2001). The norms make sure that the results measured by the rain gauge are accurate and not affected by the surrounding area like high buildings, which can cause rain shadows.

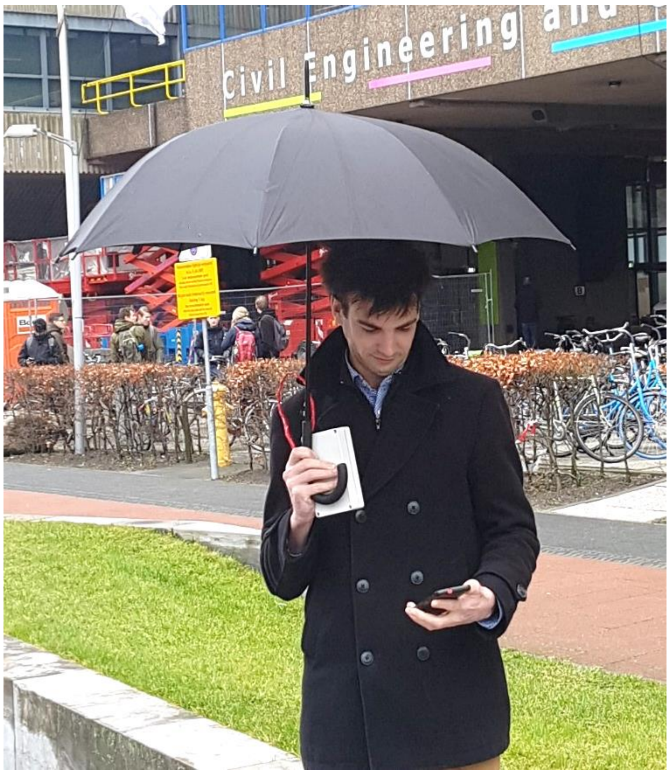


Figure 1: The author testing the umbrella in an urban area. The data measured are send to an application on the mobile phone.

Because of these norm, all official recording rain gauges within the Netherlands are placed in rural areas and barely placed in urban areas.

As most of humanity live in cities, and due to an increase in extreme rain events and associated flooding, there is an upcoming need for recording rain gauges in urban areas. However, high-density buildings make it nearly impossible to get accurate rainfall data, due to the beforementioned rain shadows. Beside this, measurements need to have a very high space-time resolution to be applicable for urban hydrology. A possible solution to this problem is to measure rainfall with a lot of rain gauges causing the resolution to become very small and thereby errors will be minimalised. Under laboratory conditions several rain gauges have been tested (Lanza and Stagi, 2008) and found that the gauges complied with the WMO accuracy specifications of 5 percent. As setting up a high-density rain gauge system is costly, most urban rainfall data collection is done with use of radar images. However, the data is not yet sufficiently reliable for hydrological application in urban areas (Emmanuel, 2011). Alternatively, the use of microwave links has a high potential for use in urban areas (Upton, 2004).

In this paper a low-budget solution is presented: an umbrella as a mobile acoustic rain gauge using low-cost electronics (Figure 1). The idea is using the canvas of an umbrella as an acoustic disdrometer (Hut, 2013). Using umbrellas as rain gauges gives the possibility to measure rainfall at a high space-time resolution when several umbrellas are equipped with the necessary equipment. When the umbrella is equipped with Bluetooth technology, it is possible to send real-time the location and rainfall data with help of a mobile phone to an online server. This design paper focusses on three parts: first, the umbrella sends a Bluetooth signal both when opened and closed. This signal is responded by an android application. Second, the umbrella starts measuring rain intensity when the umbrella is opened till the moment it is closed again. Third, the umbrella sends at a specific time interval the results of the measurements.

In this report first, the choices for the materials used are elaborated, furthermore the electric circuit, coding and application are explained. Next, both experiment types are explained and the corresponding results are given: the laboratory experiment, mainly used to calibrate the electronics, and the field evaluation. The report is concluded with conclusions and discussion.

2. Methods and materials

The umbrella should register and share, with use of Bluetooth, whether it is opened or when it is close. When it is opened, it should measure rainfall intensities and send at a specific time interval the results of these measurements. In the following part, first the choices of the materials used in the umbrella are clarified. After this the electric circuit and the coding will be elucidated. Subsequently the application used will be explained as well as the set-up for both the laboratory experiment and the field evaluation.

For this first prototype proof of concept an umbrella with a diameter of 90 centimetre is used. At the top and the bottom of the shaft small holes are drilled through which four wires are pulled. To see whether the umbrella is open or closed, two of the wires are connected to a reed switch at the top of the shaft. At the moving part of the umbrella a small magnet is connected which will close the electric circuit through the reed switch when the umbrella is opened. Initially the Adafruits LIS3DH accelerometer would be used to measure the position of the umbrella arc. The movement of the arc would then

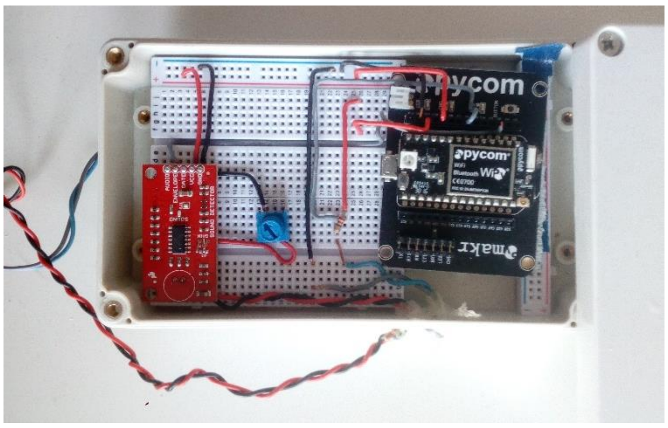


Figure 2: Watertight box with on the left the sound detector by Sparkfun connected to the piezo electric sensor on the arc of the umbrella, in the centre a potentiometer to adjust the sensitivity of the sound detector and on the right the WiPy. There is also space for a battery.

be used to detect whether the umbrella is opened or closed. However, due to compatibility problems, the accelerometer is not useable in this set-up.

Beside using the reed switch or accelerometer several other possibilities are available. For instance, placing a button in the shaft behind the lever which activates the opening of the arc. The use of a button was in this set-up not suitable, as it is quite hard to reach the area in the shaft behind the lever. Another possibility is using a hall sensor which measures nearby magnetic fields. This sensor needs a continuous stream supply to function which will cause a shorter battery life expectancy. Other solutions which encounters similar problems are using light intensity sensors (when the umbrella is closed there is no light), but when using the umbrella in the dark, the sensor will think the umbrella is still closed; a distance sensor who measures the distance between the shaft and the arc; a temperature sensor in the handle which measures a temperature difference when someone holds the handle or the use of tilt switches on the arc and the shaft: when one of the two has a different position in relation to the other, the arc is open. As the other solutions are not very practical, a reed switch is used.

The WiPy 2.0 (powered by a rechargeable 2500mAh lithium battery) is used as a low-cost CPU. This Internet of Things development platform runs on MicroPython and has an internal Bluetooth radio allowing information to be sent to a nearby device (through Classic Bluetooth or Bluetooth Low Energy, BLE) (Pycom, 2020). The Bluetooth radio will be used in its Low Energy form acting like a beacon. As the WiPy only has a limited character length for making it act like a beacon, there is no space for additional information to send with the basics like its name and MAC address. For that reason, the name of the WiPy will be changed into the value you want to send. The application used to collect the data send by the WiPy, saves the name of the WiPy on a predetermined location. The same data is also saved on a SD card plugged in the Pycom Expansion Board.

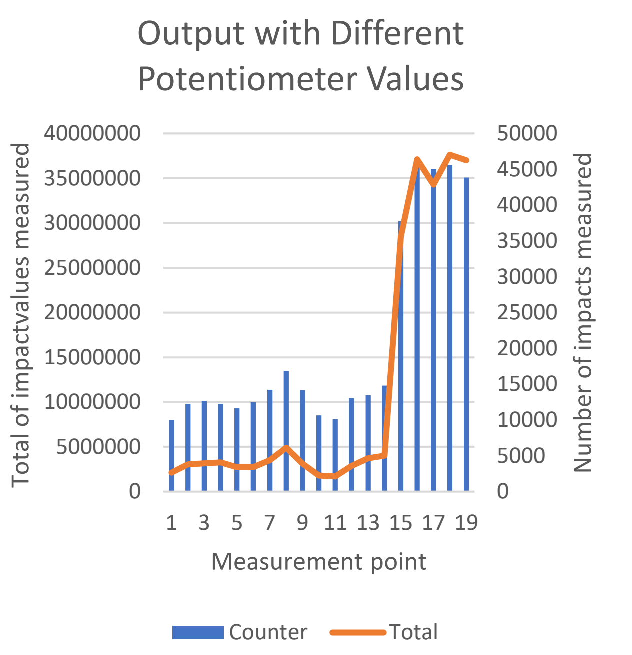


Figure 3: Laboratory experiment with different potentiometer values. As the electrical potential changes, the output of the sound detector change.

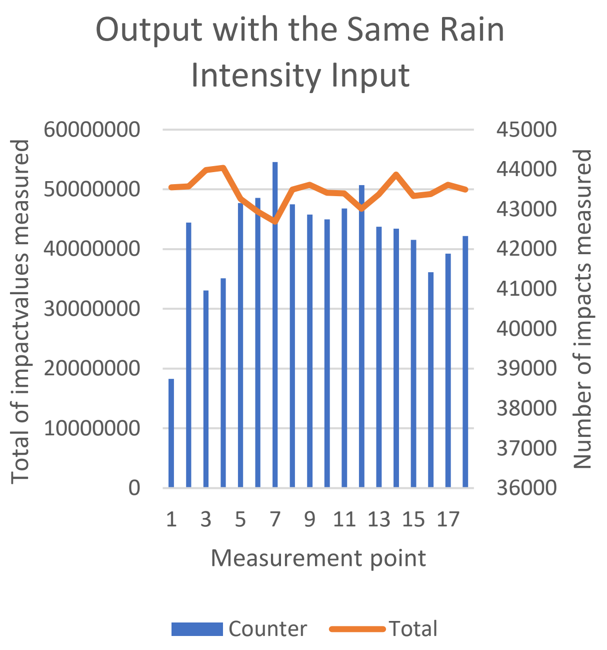


Figure 4: Laboratory experiment with the same rainfall intensity. The measured number of peaks vary strongly over time.

The pins of the WiPy need additional attention as some pins only act as input pins and the others can also act as a PWM (Puls-width modulation) pin, which can work with analog signals as output as well as acting as a input pin. When the WiPy is connected to a REPL (Read–Eval–Print Loop) console, the output of these PWM pins act in a different way, then when it is disconnected to this console and only connected to an external battery. This leads to different observed values. For that reason, it is necessary to connect the reed switch and the sound sensor to the input-only pins to make sure the same analog value is red in both situations.

As all electronics are used outdoors and during rain events, the set-up needs to be weather proof as much as possible. Therefore, the main electronics like the WiPy, battery and the sound detector, are put in a watertight box as can be seen in Figure 2.

To measure the raindrops falling on the arc of the umbrella, a piezoelectric pressure sensor is used. This sensor detects vibrations or knocking and has a voltage as output. To read out the voltage caused by the piezo sensor, a SparkFun Sound Detector is used (Sparkfun, 2020). This sound detector has as input a (sinusoidal) voltage signal from the piezoelectric sensor and turns this signal in three different outputs: an audio output, a binary indication of the presence of sound and an analog representation of its amplitude. In this application, the third output signal (the envelope output) is used. With use of this signal output, a counter of the number of droplets can be created and the amplitude of the signal caused by an individual droplet can be found.

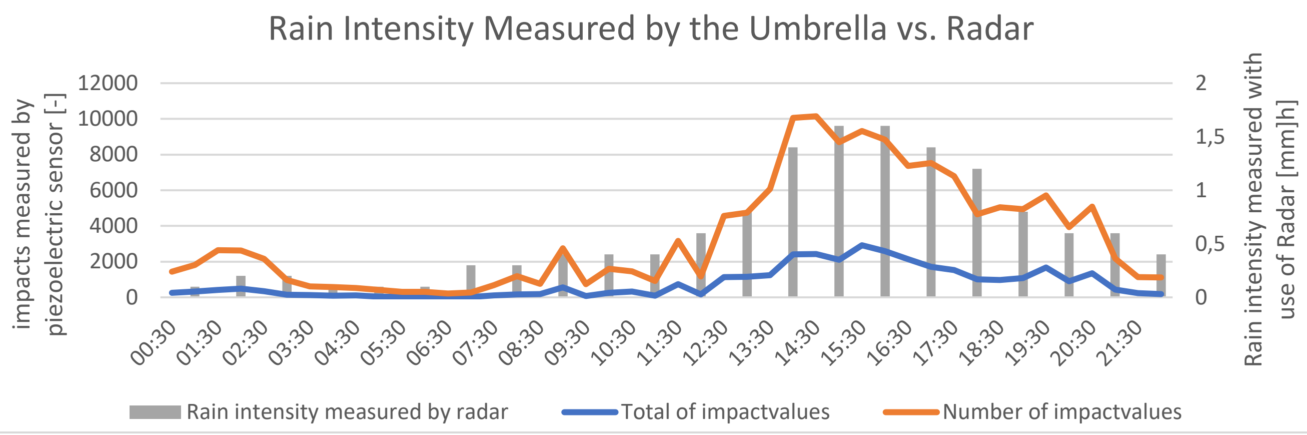
An elaboration of this coding is given in paragraph 2.2. To adjust the sensitivity (gain) of the sound detector, a potentiometer is connected to it. To calibrate the potentiometer and to see the effect of the sensor, a laboratory experiment is executed as will be elaborated in paragraph 2.4.

2.1. Circuit

To resist the wet weather conditions, the electronics are placed in a box. The main parts which needs weather protection are the WiPy, the battery and the sound detector. All electronics are connected to the WiPy which not only acts as signal processor, but also as power supplier for the reed switch and the sound sensor. To see whether the umbrella is open or closed, the signal over the reed switch is read out over pin 19 (G6), the sound detector is connected to three points: the VVC is connected to pin 22 (G9), the Envelope, which measures the amplitude of the signal, is connected to pin 15 (G0) and the ground of the sensor is connected to GND. In this way, the sound detector will only work when pin 22 is activated. Table 1 gives a complete overview of the wire connections.

Table 1: Wire connections between the WiPy, sound detector and the reed switch

Sensor	**WiPy**
Sound detector	
GND	GND
VCC	Pin 22 (G9)
Envelope	Pin 15 (G0)
Reed switch	
+	3V3
-	GND
Analog signal	Pin 19 (G6)



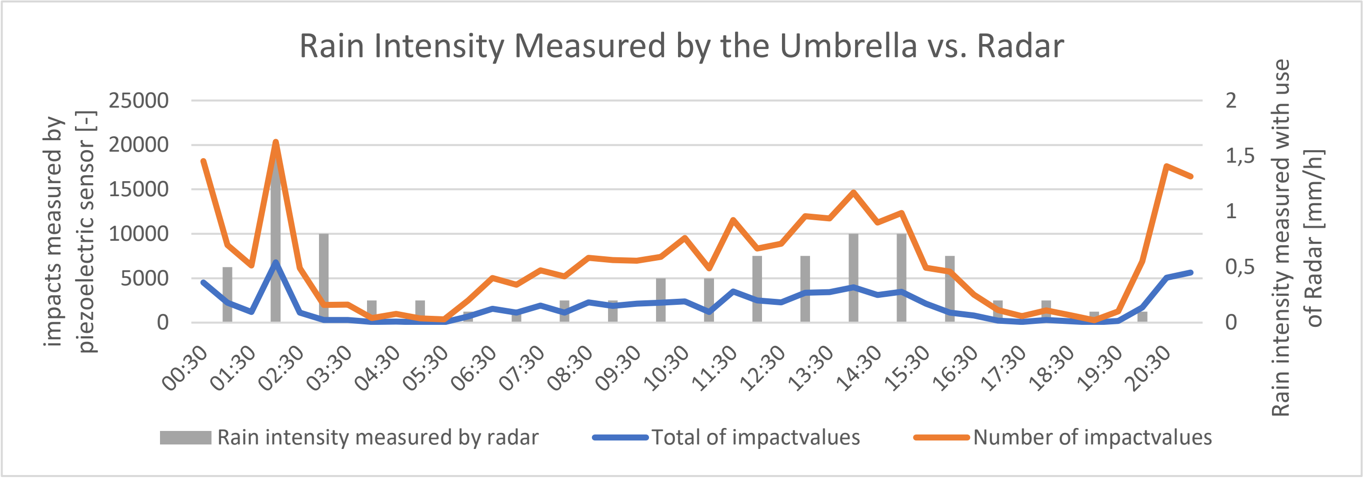


Figure 5: Results of the field evaluations during different rain events.

2.2. Coding

The code used to control the WiPy, is written in MicroPython. The main goals of the code are to detect whether the umbrella is open or not and, if opened to collect rain intensity data and send its results over Bluetooth.
To achieve this, first the WiPy reads the analog value on pin 19 (connected to the reed switch) to know whether the umbrella is open (the reed switch is near a magnet and pin 19 gives a low value) or the umbrella is closed (the analog value measured over pin 19 is high). When the umbrella status is changed, the Bluetooth module is activated and sends for 15 seconds respectively ‘HelloWorld’ or ‘ByeWorld’.
When opened, directly a 30 seconds timer is started and the analog values measured over pin 22, derived from the sound detector, are processed. For every value it is considered whether this value is higher than the previous one. If this statement becomes false, a peak has reached, which indicates the maximum of a raindrop-impact. Every maximum is added to the previous ones to save as many of the limited internal memory space available as possible. Also the amount of maxima is counted.
When the 30 seconds are over (a 30 seconds time interval is chosen to make it easier to filter extreme or inconsequent values afterwards), the sum of the maxima is send over Bluetooth. Beside this, the sum of the maxima together with the number of maxima (counter) are added to arrays, which are saved on the SD card when the umbrella is closed again. This is used as a back-up system when the transfer over Bluetooth fails. The full coding can be found in appendix A.

2.3. Application

To share the data collected by the umbrella an application on a smartphone stores the data on a predetermined location online. Every 10 seconds the application searches for pre-set MAC addresses and shouts these values to its server. Together with the sum of the maxima, the location and time of the phone is send, so it is known where and at what time the measurements are done.

2.4. Laboratory experiment

To see the effect of the potentiometer on the output of the sound sensor and to see how the umbrella reacts to water droplet impacts generally, the umbrella is tested by placing it underneath a shower. To approach rainfall as closely as possible, the showerhead is pointed upwards, so the shower spray will first go up and eventually fall scattered. To minimalize the number of drops, the showerhead is partly taped so only three nozzles are spraying water.

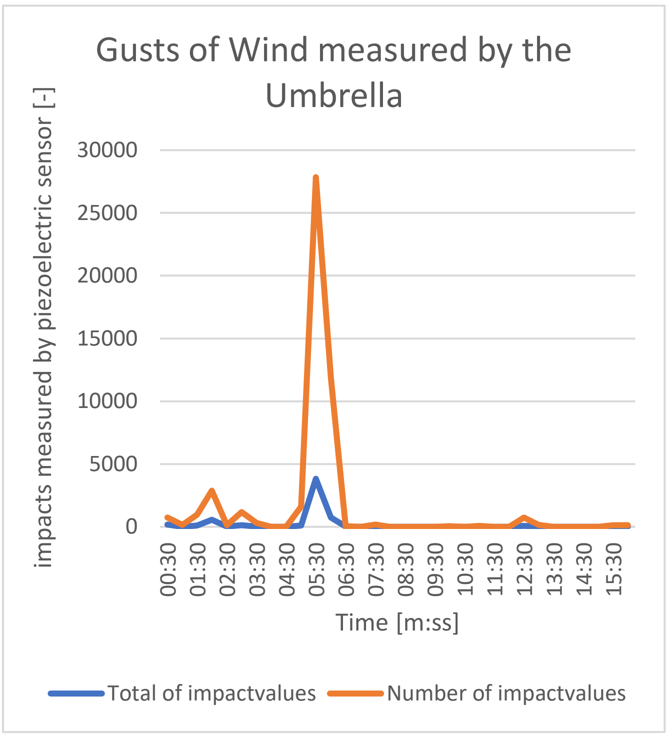


Figure 6: Graph showing the impact of gusts of wind on the results measured by the umbrella. The impact of the wind is strongly noticeable

2.5. Field evaluation

To measure the effectiveness of the umbrella in an urban area, it is tested in the field. At different locations, both windy as less windy, rainfall events are measured. As explained previously, every 30 seconds the total measured peaks and the total impact value is send over Bluetooth. At the end the results measured by the umbrella are compared to data obtained by radar. The rain intensities obtained by radar is only available time-space resolution of 5 minutes. To make the data suitable for comparison with the umbrella results, the radar data are interpolated to one a time-space resolution of one minute.

3. Results

3.1. Laboratory experiment

As elucidated in paragraph 2.4, the umbrella is placed underneath a shower head, to measure the effect of the potentiometer, it is placed in three different levels (full electrical potential, half electrical potential and minimum electrical potential) and lastly completely removed. In Figure 3 the results of this experiment are shown. The bars give the number of impact peaks measured and the line represents the total of these impact values.

It is clear to see the effect of using a potentiometer added to the sound detector. Using a potentiometer is needed as when it is left out in the circuit, the gain is at such a high level, every small vibration, which can have other origins than the water droplets, is encountered and registered.

In Figure 4 the output data during a consequent rain intensity is showed. From this graph can be conclused that even during the same rain intensity, the output data still various allot. The difference between the utmost value and the mean of al values can be up to 10%. This can be caused by inconsistency in the water droplets falling on the arc due to pressure differences, or because the gain of the sound sensor is still too high and additional vibrations are still measured. For the field evaluation the potentiometer is set in its full electrical potential.

3.2. Field evaluation

When the umbrella is opened, vibration caused by this movement are recorded by the piezoelectric sensor, so the first 30 seconds of measurements are not reliable for further use. Furthermore, gusts of wind cause vibrations of the umbrella arc which affects the results. To know the impact of a gust of wind, the umbrella is tested during a windy, but dry period.

The results of this test is plotted in the graph of Figure 6. As the measured values due to vibrations caused by gusts of wind are significant, the impact of the wind is clearly not negligible. To avoid that measurements during rain events are affected by gusts of wind, the locations of the field evaluation are chosen to be as windless as possible without staying in the rain shadow of for example a building which will influence the results as well.

The results of the experiments are found in the in graphs of Figure 5. In each graph two lines and a set of bars are plotted. The bars present the rainfall as measured with radar at the location. These quantities are compared with the measured results found with the umbrella. The upper orange line presents the number of peaks measured by the piezo electric sensor. The lower blue line presents the sum of the amplitude values measured at these peaks. In this way, it should be possible to say how many droplets fell on the area surrounding the sensor and the size of the droplets as it is assumed that bigger droplets have a greater impact on the umbrella than smaller droplets.

It can be clearly seen that when the rain intensity increases, the counter and total of values are increasing. However, with low rain intensities, up to 0.4 mm per hour, the measurements are quite variable. Probably because the droplets have not enough impact strength. From 0.6 mm per hour on, the lines clearly rise and follows the pattern of the rain intensities as measured by the radar.

4. Conclusion and discussion

The umbrella works as a mobile acoustic rain gauge during the field evaluation. This prototype proof of concept shows that umbrellas, used in urban areas, can be equipped with low cost electronics and can be used as a rain gauge. The results gathered by the umbrella set-up can be send using Bluetooth to a mobile device which can send it to the server online. The data can thereafter be used for urban water management and the validation of satellite data.

Using low-cost electronics: a piezoelectric sensor, a reed switch, a Sparkfun sound detector and a Pycom WiPy laboratory experiments and field evaluations are done. During the laboratory experiments it has become clear that the gain of the sound detector has a major influence on the number of peaks encountered. Decreasing the gain leads to a less sensitive system.

It also becomes clear that the collected data show many highs and lows for the same rain intensity. This can be due to the way peaks are measured, as a peak is registered as a peak when the following data point is lower than the previous one. This lower value can be an incident and still be part of a rising line towards the real peak.

During the field evaluation it has become clear that the impact of the wind can’t be neglected as the number of impact values strongly rises. However, the impact on the total of the impact values is clearly less influential. When measuring rain events in areas with less wind, the results of the field evaluation are promising as the values measured by the umbrella follows the rain intensities as measured by radar.

If the umbrella is further developed, the following improvements should be made. First, the impact of the potentiometer on the sound detector should be further examined to know what impact lowering the gain has on the number of impacts measured. Furthermore, one should try to achieve a more stable signal output during similar rain intensities by adjusting the gain or using more piezoelectric sensors on the arc of the umbrella. Lastly the limited internal memory capacity of the WiPy should be considered as individual data points can’t be extracted from the results measured.

5. References

Emmanuel, I., Andrieu, H., and Tabary, P., Evaluation of the new French operational weather radar product for the field of urban hydrology, *Elsevier*, 2011

Hut, R.W., New Observational Tools and Datasources for Hydrology, *TU Delft*, 2013

KNMI, Handboek Waarnemingen – Hoofdstuk 6: Neerslag, *Koninklijk Nederlands Meteorologisch Instituut,* 2001

Lanza, L.G., and Stagi, L., Certified accuracy of rainfall data as a standard requirement in scientific investigations, *Advances in Geosciences*, 2008

Pycom: WiPy 2.0, https://pycom.io/wp-content/uploads/2018/08/wipy2-specsheet.pdf, last access: 14 April 2020

Sparkfun: Sound Detector SEN-12642, https://www.sparkfun.com/products/12642, last access: 14 April 2020

Upton, G.J.G., Holt, A.R., Cummings, R.J., Rahimi, A.R., and Goddard, J.W.F., Microwave links: The future for urban rainfall measurement?, *Elsevier*, 2004

Appendix A: Coding

```python
import pycom                                            # import all nessecary librarys
import time
import machine
import math
from machine import SD
from machine import Timer
from network import Bluetooth                           # MAC-address: 24:0A:C4:00:A5:B0

# pins: Reedsensor: pin 19 (G6), Piezosensor - VVC: pin 22 (G9), Envelope: pin 15 (G0), GND: GND

pycom.heartbeat(False)                                  # turns flashlight off
pycom.rgbled(0xf0000)                                   # turns LED to red

High = 2500                                             # threshold reed sensor

sd = SD()
os.mount(sd, '/sd')                                     # find SD-card

bluetooth = Bluetooth()                                 # bluetooth
adc = machine.ADC()                                     # analog to digital reader
apin_reed = adc.channel(pin = 'P19')                    # reads analog signal on pin 19 (G6)

class BLE:                                              # use Bluetooth for sending information
    def __init__(self):
        print(' ')
        print('Bluetooth will send:', str(self.name))   # for testing: print the information to be send
        bluetooth.set_advertisement(name = str(self.name), service_uuid = b'1123581321345589', service_data = 'FurbsTest')
        print('Bluetooth activated')
        bluetooth.advertise(True)                       # start advertising with values bluetooth.set_advertisement()
        pycom.rgbled(0xf)                               # blue LED to indicate advertisement turned on

class StopBLE:                                          # timer to stop bluetooth advertisement (counting on background)
    def __init__(self):
        self.seconds = 0                                # set timer to zero
        self.__alarm = Timer.Alarm(self._seconds_handler, 1, periodic = True)
                                                        # start timer and measure every second
    def _seconds_handler(self, alarm):
        self.seconds += 1
        if self.seconds == self.stop:
            bluetooth.advertise(False)                  # stop bluetooth advertisement
            print(' ')
            print(self.seconds, 'seconds passed #BLE')
            print('Bluetooth disabled')
            pycom.rgbled(0xf00)                         # LED to green
            alarm.cancel()                              # stop counting after 'self.seconds' seconds


class Minute:                                           # alarm (counting on background)
    def __init__(self):
        self.time = 60                                  # set initial time to 60 seconds
        self.stop = False                               # initial value, will change when umbrella is closed
        self.seconds = 0                                # initial value
        self.__alarm = Timer.Alarm(self._seconds_handler, 1, periodic = True)
                                                        # alarm; steps of 1 second
    def _seconds_handler(self, alarm):
        if self.seconds == 1:                           # for testing: print new cycle is started
```

```python
            print(' ')
            print('new Alarm cylce started')
        self.seconds += 1                               # +1 seconds
        if self.seconds == self.time - 10:              # for testing: print 10 seconds left till end cycle
            print(' ')
            print('10 seconds left #Alarm')
        if self.seconds == self.time:                   # if "self.time" seconds has passed, print the elapsed time
            print(' ')
            print(self.seconds, 'seconds passed #Alarm')
        if self.seconds == self.time + 1:               # restart counting at 1 second
            self.seconds = 1
        if self.stop == True:                           # when umbrella is closed, stop counting
            print(' ')
            print('Alarm cancelled')
            alarm.cancel()                              # stop alarm

while True:
    """Start"""
    pycom.rgbled(0xf0000)
    active = False                                      # startpoint to see whether DAC and ADC pins are activated
    counter = 0                                         # reset counter measurements piezo sensor to zero
    t2 = 0                                              # reset counter averages piezo sensor to zero

    total = 0                                           # reset total to zero
    timer = False                                       # if True: Alarm is activated
    send = False                                        # if True: Bluetooth is activated
    rising = False                                      # if True: running toward maximum
    val_max = 0                                         # startpoint maximum value piezo sensor

    lst1 = []                                           # lists to print on SD (when umbrella is closed)
    lst2 = []
    lst3 = []
    lst4 = []

    val_reed = apin_reed()                              # read analog signal reed switch (when zero: umbrella is open)

    if val_reed <= High:                                # if reedswitch near magnet (value < High), umbrella open: start measurements
        pycom.rgbled(0xf00)                             # turn LED to green
        print('Umbrella open, initiating bluetooth')
        BLE.name = 'HelloWorld'                         # send HelloWorld to show the umbrella is open
        StopBLE.stop = 15                               # send BLE signal for 15 seconds
        ble = BLE()                                     # start up class 'BLE'
        stopBLE = StopBLE()                             # start counter to stop BLE signal

        lst1.append('HelloWorld')                       # add HelloWorld to lists
        lst2.append('HelloWorld')
        lst3.append('HelloWorld')
        lst4.append('HelloWorld')

        while True:                                     # while umbrella is open:
            if active == False:                         # if first measurement loop:
                apin_sound = adc.channel(pin = 'P15')   # use pin 15 (G0) as analog-digital reader (Piezosensor)
                dac_sound = machine.DAC('P22')          # use pin 22 (G9) as DAC
                dac_sound.write(1)                      # maximum voltage over pin 22 (3V3)
                active = True                           # return pins activated

            if active == True:                          # loop this sequence:
```

```python
                        if timer == False:                      # if alarm is not activate, activate alarm (class Minute)
                            minute = Minute()                   # call class Minute
                            minute.time = 30                    # use a specific timespan in which values will be send over BLE (standard 60 seconds)
                            timer = True                        # return alarm is activated

                            counter = 0
                            total = 0

                        if minute.seconds == minute.time and send == False:
                                                                # when "minute.time" seconds have passed and BLE is not activated:
                            send = True                         # indicates BLE is activated and won't redo this loop

                            if counter > 1:                     # if rain is detected:
                                gem = total / counter           # calculate average value per maximum
                                tt = total / minute.time        # calculate average value per second
                                var = math.sqrt(float(S) / (counter - 1))
                                                                # calculate variance per maximum
                                print('total       : ', total)
                                print('counter     : ', counter)
                                print('S:', S,'M1:', M1,'M2:', M2)
                                print('variance    : ', var)
                                print('average     : ', gem)
                                print('total/time  : ', tt)

                                value1 = tt/1000                # save values in lists
                                value2 = gem
                                value3 = total
                                value4 = counter
                            else:
                                value1 = 'NoRain'               # save 'NoRain' in lists
                                value2 = 'NoRain'
                                value3 = 'NoRain'
                                value4 = 'NoRain'

                            t2 += 1                             # new cycle
                            print('measurement : ', t2)

                            BLE.name = value1                   # send the average value per second
                            StopBLE.stop = 15                   # send BLE signal for 15 seconds
                            ble = BLE()                         # start up BLE
                            stopBLE = StopBLE()                 # stop BLE signal

                            counter = 0                         # reset counter measurements Piezosensor to zero
                            total = 0                           # reset total to zero

                            lst1.append(value1)                 # save values in lists (SD)
                            lst2.append(value2)
                            lst3.append(value3)
                            lst4.append(value4)

                        if minute.seconds == 1:                 # reset "send" to False, so when a new timespan has passed, BLE can be activated again
                            send = False

                        val_sound = apin_sound()                # read value Piezosensor over P15

                        if val_sound > val_max:                 # if new value is larger than previous: save new value as maximum
                            val_max = val_sound
                            rising = True                       # indicate values are rising
```

```
170.
171.                       if val_sound <= val_max and rising == True:
172.                                                               # if new value is smaller than previous and sequence was rising: previous value was maximum
173.                           counter += 1                        # add 1 to counter
174.                           total += val_max                    # add maximum to total
175.                           if counter == 1:
176.                               M2 = val_max                    # set initial value to val_max (variance)
177.                               S = 0                           # set initial value to 0 (variance)
178.                           else:                               # calculate new values to determine variance at the end of a cycle
179.                               M1 = M2
180.                               M2 = M1 + (val_max - M1)/counter
181.                               S = S + (val_max - M1) * (val_max - M2)
182.                           rising = False                      # maaximum is reached, values are dropping
183.
184.                       if val_sound <= val_max and rising == False:
185.                                                               # if new value is smaller than previous value and trend is dropping: save new value as maximum
186.                           val_max = val_sound
187.
188.                       val_reed = apin_reed()                  # read value reedswitch over P19 (umbrella open/closed)
189.
190.                       if val_reed >= High:                    # if reedvalue is high, umbrella is closed: stop measuring
191.
192.                           minute.stop = True                  # stop alarm
193.                           bluetooth.advertise(False)          # if sending data over BLE; stop it
194.                           print('umbrella closed, activate bluetooth')
195.                           dac_sound.deinit()                  # turn of DAC over pin 22
196.
197.                           BLE.name = 'ByeWorld'               # send ByeWorld to show the umbrella is closed
198.                           StopBLE.stop = 15                   # send BLE signal for 10 seconds
199.                           ble = BLE()                         # start up BLE
200.                           stopBLE = StopBLE()                 # stop BLE signal
201.
202.                           lst1.append('ByeWorld')             # add 'ByeWorld' to lists
203.                           lst2.append('ByeWorld')
204.                           lst3.append('ByeWorld')
205.                           lst4.append('ByeWorld')
206.
207.                           os.listdir('/sd')
208.                           f = open('/sd/tt2.txt', 'a')        # print lists to SD card
209.                           f.write(str(lst1))
210.                           f.close()
211.                           f = open('/sd/average2.txt', 'a')
212.                           f.write(str(lst2))
213.                           f.close()
214.                           f = open('/sd/total2.txt', 'a')
215.                           f.write(str(lst3))
216.                           f.close()
217.                           f = open('/sd/counter2.txt', 'a')
218.                           f.write(str(lst4))
219.                           f.close()
220.
221.                           print('lists send to SD card')
222.
223.                           break                               # break and return to """Start""" (line 72)
```