ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta Dopravní

ZPRACOVÁNÍ DAT

11ZDA

Semestrální práce

17.11.2017

Bc. Lukáš Posekaný

Bc. Michal Růžička

Zadání:

Vytvoření systému schopného zaznamenávat bluetooth komunikaci vozidel a uložit informaci (čas, adresa) vztahujících se k projíždějícím vozidlům.

Ověřit fungování systému měřením v terénu.

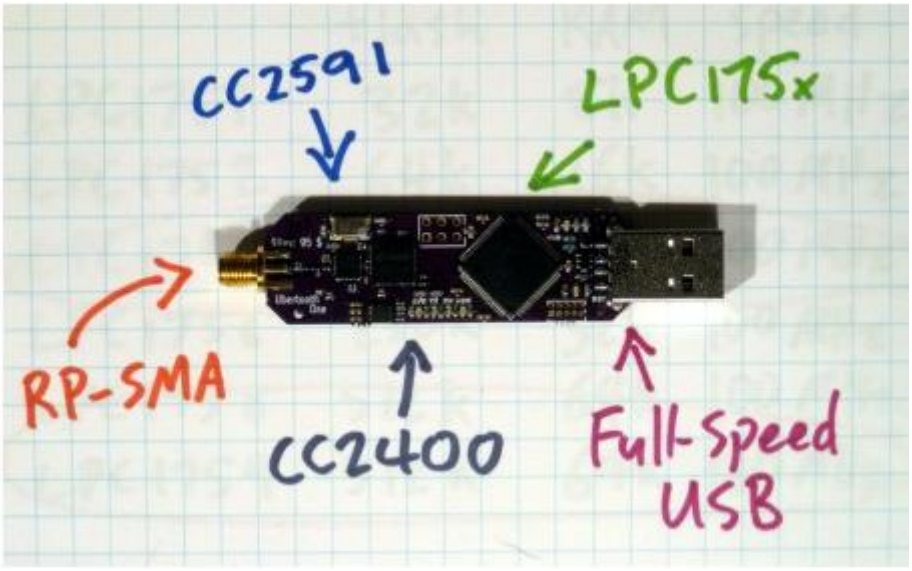
Naměřená data zhodnotit a posoudit účinnost bluetooth detektorů

Zvolení vhodného bluetooth rozhraní:

Pro možnost odchytávání bluetooth komunikace byl zvolen přístroj Ubertooth One. Toto zařízení je speciálně navrženo pro experimentální testování s bezdrátovými technologiemi a dle popsaných parametrů je pro tento projekt zcela vyhovující.

Části Ubertooth One:

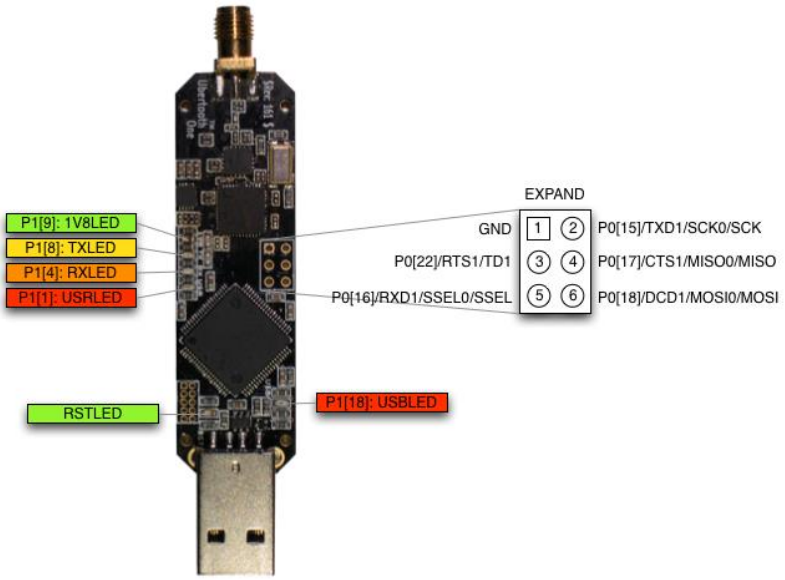
- RP-SMA RF connector
- CC2591 RF front end
- CC2400 wireless transceiver
- LPC175x ARM Cortex-M3 (Full-Speed USB microcontroller)
- USB A plug



Obrázek1.: Části Ubertooth One

Ubertooth parametry:

- 2.4 GHz příjem a vysílání
- Vysílací síla a citlivost srovnatelné s Class 1 Bluetooth
- Standard Cortex Debug Connector (10-pin 50-mil JTAG)
- In-System Programming (ISP) seriál connector
- Expansion connection: pro vnitřní komunikaci
- 6 indikačních LED



Obrázek 2.: Diody a rozhraní Ubertooth One

Zařízení je určeno především pro propojení s operačnímy systémemy Linux a projekt bude tedy navržen pro tento systém.

Ubertooth One lze připojit k počítači přes USB rozhraní a po jeho připojení je třeba nainstalovat potřebné balíčky. Instalace je znázorněna na následujících řádcích.

Předpoklady systému pro instalaci ostatních balíčků

Pro úspěšnou instalaci je třeba mít již nainstalované některé balíčky jako cmake, make, gcc, g++ ... Lze potřebné prostředí nainstalovat pomocí příkazu:

```
sudo apt-get install cmake libusb-1.0-0-dev make gcc g++ libbluetooth-dev \

pkg-config libpcap-dev python-numpy python-pyside python-qt4
```

Instalace lbbtbb

Další knihovnou sloužící k dekódování Bluetooth paketů je libbtbb. Instalaci lze provést následujícími příkazy:

```
wget https://github.com/greatscottgadgets/libbtbb/archive/2017-03-R2.tar.gz
-O libbtbb-2017-03-R2.tar.gz

tar xf libbtbb-2017-03-R2.tar.gz

cd libbtbb-2017-03-R2

mkdir build

cd build

cmake ..

make

sudo make install
```

Ubertooth tools

Poslední knihovou je knihovna Ubertooth, kterou lze nainstalovat příkazy:

```
wget https://github.com/greatscottgadgets/ubertooth/releases/download/2017-
03-R2/ubertooth-2017-03-R2.tar.xz -O ubertooth-2017-03-R2.tar.xz

tar xf ubertooth-2017-03-R2.tar.xz

cd ubertooth-2017-03-R2/host

mkdir build

cd build

cmake ..

make

sudo make install
```

Zobrazení spektra pomocí Ubertooth One:

Po zapojení Ubertooth One do USB portu lze systém otestovat skenováním spektra. To lze provést příkazem:

```
ubertooth-specan-ui
```

Odchytávání paketů pomocí Ubertooth One:

Pro odchytávání a zobrazování komunikace slouží několik příkazů:

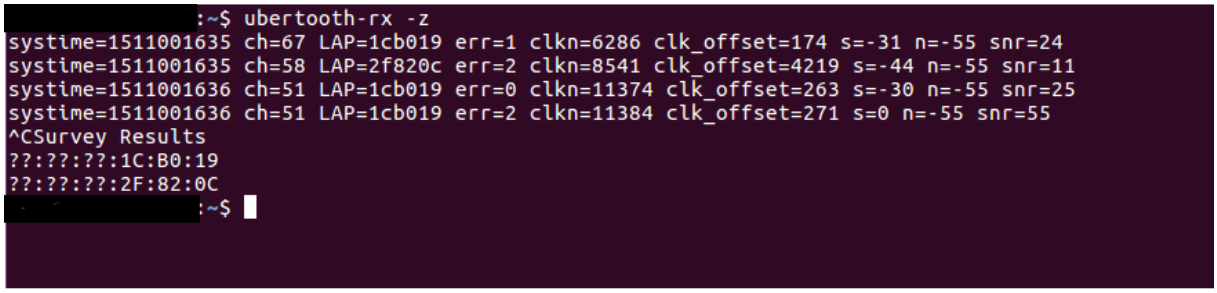
```
ubertooth-rx

ubertooth-btle -p
```

Tyto funkce mají dále mnoho přídavných parametrů pro upřesnění odchytávání paketů. Pro potřebu detektoru bude třeba pouze zaznamenat dobu zachycení paketu a část adresy (LAP). K tomu slouží následující příkaz:

```
ubertooth-rx -z
```

Na obrázku 3 je je vidět výstup ze zmíněného příkazu. Pro naše potřeby bude potřeba pracovat se sloupcem číslo 0. A 2. (tedy 1. a 3.)



Obrázek 3.: Výstup z Ubertooth One pro příkaz ubertooth-rx -z

Systém Bluetooth detektoru:

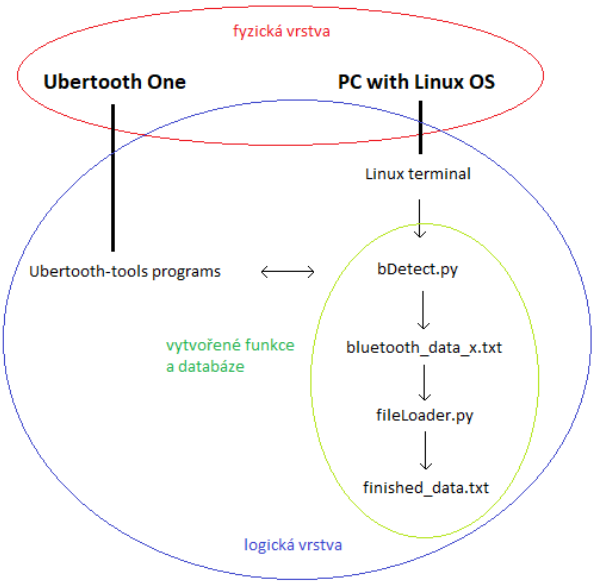
Fyzické schéma systému:

Fyzická část systému se skládá z Ubertooth One a PC s Linuxovým operačním systémem. Ubertooth One obsahuje veškeré potřebné komponenty pro zachycení elektro magnetických vln jako jsou anténa, AD převodník a procesor pro primární zpracování dat. K PC lze Ubertooth One připojit pomocí USB portu. Na PC vyhodnocující data nejsou kladeny žádné speciální nároky. Musí mít volné USB porty a některou z Linuxových distribucí jako operační systém, případně virtuální box pro spuštění virtuálního operačního systému. Fyzické schéma je znázorněno na částí Obrázku 4.

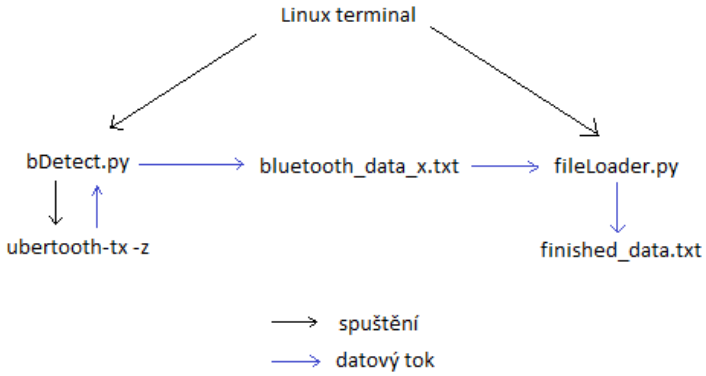
Logické schéma systému:

Logické schéma systému je dělené na Linuxový terminál, Ubertooth-tools knihovnu a vytvořené programy a databáze. Linuxový terminál slouží ke spuštění vytvořeného programu bDetect.py. Tento program spouští pomocí volání os příkaz ubertooth-rx -z z knihovny Ubertooth-tools která programu vrací prostřednictvím os data o jednotlivých odchycených paketech. Tato data program zpracovává a zaznamenává do textového souboru bluetooth_data_x.txt. Z tohoto souboru data lze po měření dále získat a provést nad nimi různé úpravy (třídění, počítání četnosti, …) pomocí programu fileLoader.py, který data buď tiskne nebo ukládá do souboru finished_data.txt.

Logické schéma je znázorněno na části Obrázku 4., Spouštění funkcí a datové toky jsou znázorněny na Obrázku 5.



Obrázek 4.: Schéma systému bluetooth detektoru



Obrázek 5.: Spouštění a datové toky systému

Požadované funkce systému:

Pro systém byly vyhrazeny tyto funkční nároky:

- Automaticky spustit ubertooth-rx -z
- Zaznamenávat data z Ubertooth One v reálném čase
- Z dat extrahovat požadované sloupce (čas, adresa)
- Data dále upravit a uložit:
    - V případě více záznamů od jednoho zařízení zapsat pouze první záznam (pokud vozidlo stojí v koloně, …)
    - Při každém měření vytvořit soubor s novým pořadovým číslem
    - Data ukládat průběžně aby v případě náhlého vypnutí systému zůstala data zachována

Veškerý kód programů viz Příloha 1. a Příloha 2.

Způsob ověření systému

Po otestování systému v laboratorních podmínkách bylo naplánováno měření v terénu pro otestování v reálných podmínkách

Měření v terénu:

Pro ověření systému proběhlo měření za použití vytvořeného detektoru a pro ověření byl pořízen video záznam. Záznam byl následně zpracován a byly zjištěny celkové součty stejně jako součty za jednotlivé časové intervaly

Poloha a doba měření:

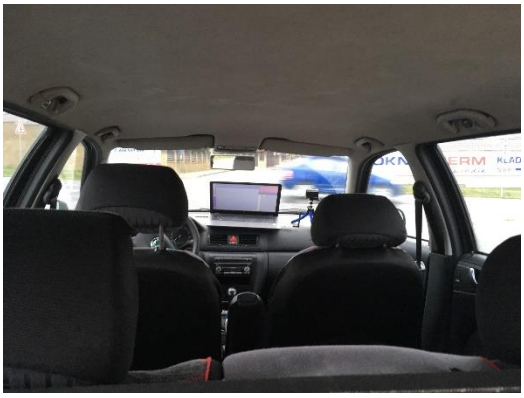
Jelikož není za potřebí získávat informace pro porozumění dopravního proudu (průběh denní intenzity, ...) není zvolení místa a času pro práci nijak významné. Požadavkem bylo, aby byla intenzita dostatečně vysoká pro získání dostatečného množství dat pro ohodnocení účinnosti systému.

Pro měření byla zvolena komunikace Berounská vedoucí z Velké dobré na Braškov a napojující se na dálnici D6, viz Obrázek 6. Stanoviště bylo zvoleno u výjezdu z obce Velká Dobrá kde je velká intenzita ovšem netvoří se zde žádné kolony, stanoviště viz Obrázek 7.

Měření proběhlo v pátek 17.11. 2017 a to od 13:11 hodin.

Obrázek 6.: Mapa znázorňující místo měření

Obrázek 7.: Stanoviště 1.o



Obrázek 8.: Stanoviště 2.o

Obrázek 9.: Stanoviště 3. o

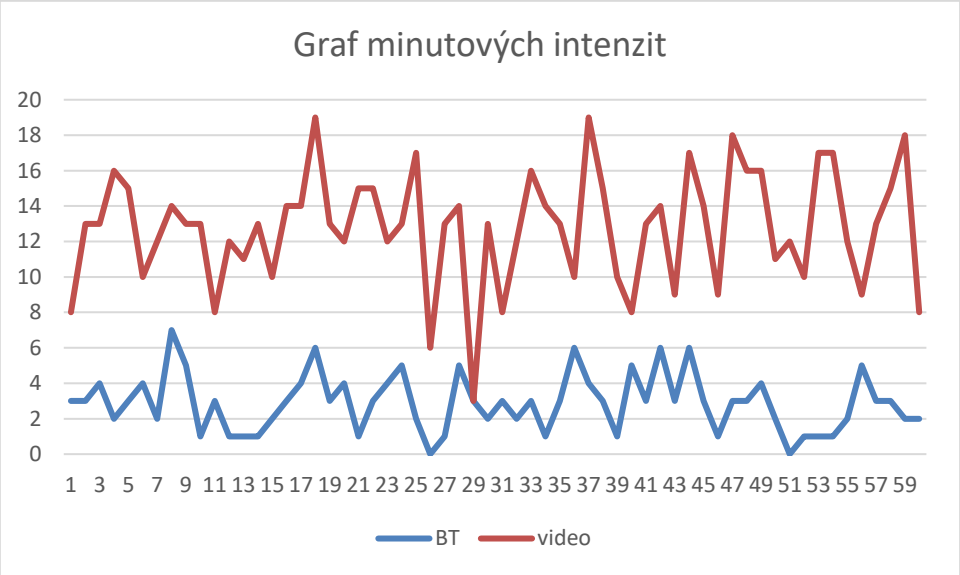
Vyhodnocení dat z měření

Pro vyhodnocení dat z bluetooth detektoru byl použit vytvořený program file.py. Výstupní data jsou pak seřazená, spočtené minutové intenzity a celkový součet vozidel. Jedním z výstupů je také četnost opakování jednotlivých adres. Výstup viz Příloha 4.

Pro kontrolu byl pořízen videozáznam celého měření. Pro vyhodnocení videozáznamu byl vytvořen program, který po stisknutí enteru uloží čas. Lze tedy z takového výstupu jednoduše získat intenzity v jednotlivých časových intervalech (60s). Kód pro vyhodnocení z vida viz Příloha 3., výsledky viz Příloha 4.

Z výsledků plyne, že 6 bluetooth adres bylo zaznamenáno 2x, ostatních 162 jednou, jedna adresa byla pro stálou přítomnost vyřazena. Intervaly jsou znázorněny na grafu 1.

Výsledný součet dat je: 767 vozidel zaznamenaných videokamerou a 173 vozidel zaznamenaných bluetooth detektorem. Budeme-li předpokládat že data z videokamery jsou zcela kompletní (tedy 100%), potom je úspěšnost bluetooth detektoru 22,5%.



Graf1.: Minutové intenzity

Závěr

Zaznamenávání vozidel pomocí bluetooth detektoru je možné, není však stále mnoho vozidel s aktivními bluetooth zařízeními. 22,5% na dopravní průzkumy postačující není, pro získání doplňkových dat však posloužit může.

Příloha1.: Kód programu bDetect.py

```
import subprocess
import time
import re
import os
from sys import exit

CWD = "/home/moja/Programming/Python/Ubertooth"
FILE_NAME_GROUP = "bluetooth_data_"

#function reat all files in path and find last one in group
#create a new file and return path to this file
def printWelcome(T, path):
	print(">>> WELCOME TO A PROGRAM FOR BLUETOOTH DETECTION <<<\n")
	print("Starting at:\t[", int((T/3600)%24), ':', int((T/60)%60), ':',
int((T)%60), ']')
	print("Save to dir: \t", path, '\n\n')

def isInt(s): #check if string represent int number
	try:
		int(s)
		return True
	except ValueError:
		return False

def newFile(dirPath, filesGroup):   #find free smallest fileName and return this
name
	dirPath = str(dirPath)
	filesGroup = str(filesGroup)
	biggestInDir = 0
	smallestFreeInDir = 0
	NewFileName = ''

	for file in os.listdir(dirPath):
		if file.endswith('.txt') and file.startswith(filesGroup):
			fileNumb = file[len(filesGroup):-4]
			if (isInt(fileNumb) and biggestInDir<int(fileNumb)):
				biggestInDir = int(fileNumb)

	smallestFreeInDir = biggestInDir+1
	NewFileName = filesGroup + str(smallestFreeInDir) + '.txt'
	return NewFileName

def writeToFile(filePath, stringToWrite):  #take filePath, open file and write
thingToWrite
	filePath = str(filePath)
	stringToWrite = str(stringToWrite)

	try:
		with open(filePath, 'a') as f:      #open file and close it no matter
how nested code exit
			f.write(stringToWrite + "\n")
	except IOError:
		print("unable to open file for writting")
		exit()
```

```
def main():

	FILE_NAME = newFile(CWD, FILE_NAME_GROUP)
	START_TIME = int(time.time())
	WAIT_TIME = 10 #repeateTime [s]
	ALREADY_LOADED_LAP = []


	printWelcome(START_TIME, FILE_NAME)

	startT = 'Starting at:\t[' + str(int((START_TIME/3600)%24)) + ':' +
str(int((START_TIME/60)%60)) + ':' + str(int((START_TIME)%60)) + ']\n'
	writeToFile(FILE_NAME, startT)


	#UBERTOOTH CAN LOAD TIME ONLY IN SECONDS => ITS NONSENSE TO CALCULATE
MILISECONDS
	#WAIT_TIME is for program to find new LAP address again after this time
	#it iterate through array and if it find LAP woth bigger time difference than
repeateTime it arease it from array and this device can be loaded again

	process = subprocess.Popen(['ubertooth-rx', '-z'], stdout=subprocess.PIPE,
bufsize=1)

	for line in iter(process.stdout.readline, b''):

		currentTime = int(time.time())
		#remove from array all expired elements
		if ALREADY_LOADED_LAP:	#if array is not empty
			for el in ALREADY_LOADED_LAP:
				if el[0]+WAIT_TIME <= currentTime:
					ALREADY_LOADED_LAP.remove(el)
		line = line.decode(encoding="utf-8")
		line = line.split(' ')


		#get parameters from whole line
		parameters = []
		for col in line:
			if re.match(r'^systime', col):
				parameters.append(col)
			elif re.match(r'^LAP', col):
				parameters.append(col)

		if parameters:#check if array is not empty and i can print it (else
it throws error and end process)
			#extract exact values from strings
			systime = parameters[0]
			systime = int(systime.split('=')[1])
			LAP = parameters[1]
			LAP = LAP.split('=')[1]

			#check if is contained in ALREADY_LOADED_LAP
			iSawIt = False
			if ALREADY_LOADED_LAP:
				for el in ALREADY_LOADED_LAP:
```

```
                                        if LAP == el[1]:
                                               iSawIt = True

                              #if not write it down
                              if (iSawIt == False):
                                     newDev = []
                                     newDev.append(systime)
                                     newDev.append(LAP)

                                     #Catched time convert
                                     CT = int(newDev[0]-START_TIME)
                                     CTs = int(CT%60);
                                     CTm = int((CT/60)%60);
                                     CTh = int((CT/3600)%24);
                                     catchedDev = str(newDev[1]) + ';' + str(CTh) + '.' +
str(CTm) + '.' + str(CTs)

                                     #print
                                     print(catchedDev)
                                     writeToFile(FILE_NAME, catchedDev)
                                     ALREADY_LOADED_LAP.append(newDev)

              process.stdout.close()
              process.wait()

main()
```

Příloha2.: Kód programu file.py

```
def readFile(fileName):
	devArray = []
	with open('f.txt', 'r') as f:
		i = 0;
		for l in f:
			if i>=1:
				break
			i = i+1;
		for line in f:
			catch = line.split(';')
			LAP = catch[0].strip()
			time = catch[1].strip()
			time = time.split('.')
			time = int(time[0])*3600+int(time[1])*60+int(time[2])
			NEW = [LAP, time]
			devArray.append(NEW)
	return devArray

def printMatrix(A):
	for el in A:
		element = ''
		for e in el:
			element = element + str(e) + '\t'
		print(element)

def writeMatrix(A, fileName):
	try:
		with open(fileName, 'a') as f:
			for el in A:
				for i in range(0,len(el)):
					if i==len(el)-1:
						f.write(str(el[i])+'\n')
					else:
						f.write(str(el[i])+';')
		return 1
	except IOError:
		print('error while reading to file')
		return 0

def order(A, index=-1): #order Array from low to high, if index is passed sorte [[]]
in order that index
	if index >= len(A[0]):
		return 0;
	elif index == -1:
		for i in range(len(A)-1, 0, -1):
			for j in range(i):
				if A[j]>A[j+1]:
					temp = A[j]
					A[j] = A[j+1]
					A[j+1] = temp
		return A
	else:
		for i in range(len(A)-1, 0, -1):
			for j in range(i):
```

```
                                if A[j][index]>A[j+1][index]:
                                        temp = A[j]
                                        A[j] = A[j+1]
                                        A[j+1] = temp
                return A

def countOccurancy(A, index=0):     #count occurancy of element related to index,
!!!other indexes get screwed
        countA = []
        firstEl = [A[0][index], 1]
        countA.append(firstEl)
        print (countA)

        for i in range(1, len(A)):
                isIN = False
                for el in countA:
                        if A[i][index] == el[index]:
                                isIN = True
                                el[len(el)-1] = el[len(el)-1]+1
                                break

                if isIN == False:
                        new = [A[i][index], 1]
                        countA.append(new)
        return countA

def countInterval(A, interval, index=1):   #count occuranvy of times in given
intervals
        start = 0
        end = start+interval
        occurancy = []
        i=0
        count = 0
        for el in A:
                jump = int((el[index]-start)/interval)     #give integer of how many
intervals the number jump cross
                if jump<1:
                        count = count+1
                while jump>=1:
                        if jump==1:
                                count = count+1
                        newInterval = [start, end, count]
                        occurancy.append(newInterval)
                        start = end
                        end = start+interval
                        count = 0
                        jump = jump-1

        return occurancy


def main():
        content = readFile('f.txt')

main()
```

Příloha 3.: Kód programu videoCount.py

```
import time

def writeToFile(filePath, stringToWrite):  #take filePath, open file and write
thingToWrite
	filePath = str(filePath)
	stringToWrite = str(stringToWrite)

	try:
		with open(filePath, 'a') as f:	#open file and close it no matter
how nested code exit
			f.write(stringToWrite + "\n")
	except IOError:
		print("unable to open file for writting")
		exit()



def main():
	START_TIME = time.time()
	FILE_PATH = "/home/moja/Programming/Python/Ubertooth/VC.txt"


	while True:
		char = input('press enter for time or E and enter for exit')
		if char=='E':
			break
		T = str(int(time.time()-START_TIME))
		writeToFile(FILE_PATH, T)

main()
```

Příloha 4.: Výstup z měření

Bluetooth	
LAP	time[s]
f6035e	21
c7dcc8	48
78cea0	78
266b6d	80
5309c0	101
551746	134
f9b6c0	156
3ffe7e	164
e13e35	168
0d9e71	215
5bc275	227
860625	253
df4f78	254
cdbcea	255
69aed2	302
0028d5	314
f036d2	334
c7b344	344
25c94f	378
03f000	396
c638c6	428
469daa	438
395480	448
dbc380	451
e6a416	455
e342af	463
ea0eee	466
7b0ab6	483
edf9fc	501
d0318e	505
4ee9f2	505
54559f	539
e3410f	596
3ffe7e	600
e13e35	606
d314e8	659
33a19d	679
d2e695	685
0cd978	809
367f68	845
f51d54	881
e5151d	909
87d275	909

Video	
#	time[s]
1	4
2	8
3	22
4	24
5	33
6	50
7	53
8	59
9	65
10	68
11	68
12	72
13	74
14	77
15	79
16	79
17	103
18	104
19	107
20	109
21	112
22	134
23	135
24	139
25	140
26	148
27	152
28	155
29	156
30	162
31	164
32	167
33	177
34	178
35	184
36	191
37	194
38	203
39	204
40	204
41	214
42	215
43	216

intervals			
interval [s]		bluetooth	video
0	60	3	8
60	120	3	13
120	180	4	13
180	240	2	16
240	300	3	15
300	360	4	10
360	420	2	12
420	480	7	14
480	540	5	13
540	600	1	13
600	660	3	8
660	720	1	12
720	780	1	11
780	840	1	13
840	900	2	10
900	960	3	14
960	1020	4	14
1020	1080	6	19
1080	1140	3	13
1140	1200	4	12
1200	1260	1	15
1260	1320	3	15
1320	1380	4	12
1380	1440	5	13
1440	1500	2	17
1500	1560	0	6
1560	1620	1	13
1620	1680	5	14
1680	1740	3	3
1740	1800	2	13
1800	1860	3	8
1860	1920	2	12
1920	1980	3	16
1980	2040	1	14
2040	2100	3	13
2100	2160	6	10
2160	2220	4	19
2220	2280	3	15
2280	2340	1	10
2340	2400	5	8
2400	2460	3	13
2460	2520	6	14
2520	2580	3	9

occurancy	
LAP	occurred
f6035e	1
c7dcc8	1
78cea0	1
266b6d	1
5309c0	1
551746	1
f9b6c0	1
0d9e71	1
5bc275	1
860625	1
df4f78	1
cdbcea	1
69aed2	1
0028d5	1
f036d2	1
c7b344	1
25c94f	1
03f000	1
c638c6	1
469daa	1
395480	1
dbc380	1
e6a416	1
e342af	1
ea0eee	1
7b0ab6	1
edf9fc	1
d0318e	1
4ee9f2	1
54559f	1
e3410f	1
d314e8	1
33a19d	1
d2e695	1
0cd978	1
367f68	1
f51d54	1
e5151d	1
87d275	1
833f26	1
a49089	1
29714d	1
46356	1

833f26	913
a49089	966
29714d	991
46356	991
02515a	1011
19a8d2	1045
406b98	1052
09f067	1053
aa8611	1055
5de2e4	1068
ebd611	1076
4b30b5	1082
3e6b30	1092
c6b79d	1100
385300	1160
84717a	1165
393f0f	1172
563307	1196
10d708	1246
5ce470	1272
537203	1307
62738	1312
20df03	1323
3f7980	1327
b5afd2	1347
4fa48f	1369
63726a	1421
463700	1426
63726a	1431
5333fa	1434
bfb114	1436
dae9b9	1442
a10a03	1458
61c7c0	1556
2bb403	1652
2bb403	1667
22ea24	1676
279560	1676
dc2d66	1678
000e67	1683
3534c2	1708
500716	1720
3b3a9b	1746
7546b3	1746
b120bc	1815
ba9aa1	1845

44	218
45	222
46	224
47	226
48	227
49	230
50	236
51	242
52	248
53	250
54	252
55	254
56	256
57	260
58	267
59	268
60	285
61	286
62	289
63	295
64	297
65	299
66	301
67	304
68	314
69	323
70	325
71	340
72	345
73	345
74	355
75	359
76	362
77	366
78	370
79	377
80	379
81	382
82	383
83	388
84	393
85	393
86	395
87	397
88	428
89	439

2580	2640	6	17
2640	2700	3	14
2700	2760	1	9
2760	2820	3	18
2820	2880	3	16
2880	2940	4	16
2940	3000	2	11
3000	3060	0	12
3060	3120	1	10
3120	3180	1	17
3180	3240	1	17
3240	3300	2	12
3300	3360	5	9
3360	3420	3	13
3420	3480	3	15
3480	3540	2	18
3540	3600	2	8
SUM		173	767

02515a	1
406b98	1
09f067	1
aa8611	1
5de2e4	1
ebd611	1
4b30b5	1
3e6b30	1
c6b79d	1
385300	1
84717a	1
393f0f	1
563307	1
10d708	1
5ce470	1
537203	1
62738	1
20df03	1
3f7980	1
b5afd2	1
4fa48f	1
463700	1
5333fa	1
bfb114	1
a10a03	1
61c7c0	1
22ea24	1
279560	1
dc2d66	1
000e67	1
3534c2	1
500716	1
3b3a9b	1
7546b3	1
b120bc	1
ba9aa1	1
7d4ed7	1
8b8aba	1
fb858c	1
4eb5ee	1
2ced73	1
c97da3	1
a7c35c	1
0099ad	1
bc59a4	1
d63f6d	1

7d4ed7	1848		90	439		c8f7aa	1
8b8aba	1904		91	441		e91da3	1
fb858c	1912		92	443		3bf816	1
4eb5ee	1939		93	446		69306b	1
2ced73	1941		94	449		3b54ad	1
c97da3	1950		95	452		d87866	1
a7c35c	1986		96	462		4f5d7f	1
0099ad	2050		97	465		bad5b8	1
bc59a4	2086		98	466		a9d169	1
d63f6d	2094		99	468		c20019	1
c8f7aa	2103		100	478		927991	1
e91da3	2104		101	479		59615f	1
3bf816	2107		102	481		230cc2	1
69306b	2116		103	483		90fe2d	1
3b54ad	2121		104	490		3e7d96	1
d87866	2142		105	499		6c6c36	1
4f5d7f	2172		106	500		2fbed7	1
bad5b8	2191		107	501		6a025b	1
a9d169	2205		108	504		27c271	1
c20019	2216		109	505		8b5559	1
927991	2222		110	512		bcfd36	1
59615f	2253		111	516		3bcdf4	1
230cc2	2267		112	522		848fbf	1
90fe2d	2318		113	524		c987ca	1
3e7d96	2345		114	536		a0228e	1
6c6c36	2352		115	541		fa8efe	1
2fbed7	2353		116	542		cbd0a9	1
6a025b	2391		117	553		3b5ad7	1
27c271	2398		118	569		3b4d9f	1
8b5559	2412		119	572		6a63f0	1
bcfd36	2427		120	574		f5166e	1
3bcdf4	2456		121	575		42ea5b	1
848fbf	2467		122	581		b5cc03	1
c987ca	2480		123	583		b1ef30	1
a0228e	2506		124	587		1b2205	1
fa8efe	2513		125	590		2aabbb	1
cbd0a9	2514		126	594		e43cd6	1
3b5ad7	2514		127	597		aabb83	1
3b4d9f	2530		128	600		0ec8db	1
6a63f0	2534		129	601		b4067b	1
f5166e	2549		130	602		a7cf1f	1
42ea5b	2582		131	606		9d9592	1
b5cc03	2583		132	608		0aeab2	1
b1ef30	2603		133	609		8ae647	1
1b2205	2614		134	636		3.38E+17	1
2aabbb	2619		135	658		ef1460	1

dae9b9	2626
e43cd6	2664
aabb83	2669
0ec8db	2676
b4067b	2679
a7cf1f	2778
9d9592	2795
0aeab2	2819
8ae647	2832
3.4E+17	2842
ef1460	2864
0b6170	2896
7520cd	2912
55fa9d	2913
6bb5ec	2933
85766f	2984
982e1d	2988
2ac8ec	3053
0cdeff	3135
1b3d0b	3212
1b1bec	3268
4bad11	3273
bc0be5	3317
19a8d2	3326
1d2578	3331
67f78d	3339
f47adb	3355
c4c2ef	3368
a77347	3370
d5b549	3373
90a31e	3438
f1cf62	3464
c98fc1	3470
d76c33	3501
7ad422	3539
3cc951	3557
2e892c	3570
54b129	3644
0d91e8	3648

136	668
137	669
138	677
139	679
140	681
141	682
142	698
143	703
144	705
145	712
146	713
147	716
148	722
149	723
150	724
151	724
152	731
153	732
154	733
155	734
156	735
157	737
158	741
159	782
160	794
161	795
162	796
163	799
164	800
165	802
166	806
167	806
168	808
169	818
170	832
171	838
172	845
173	854
174	859
175	863
176	873
177	875
178	881
179	887
180	891
181	894

0b6170	1
7520cd	1
55fa9d	1
6bb5ec	1
85766f	1
982e1d	1
2ac8ec	1
0cdeff	1
1b3d0b	1
1b1bec	1
4bad11	1
bc0be5	1
1d2578	1
67f78d	1
f47adb	1
c4c2ef	1
a77347	1
d5b549	1
90a31e	1
f1cf62	1
c98fc1	1
d76c33	1
7ad422	1
3cc951	1
2e892c	1
54b129	1
0d91e8	1
3ffe7e	2
e13e35	2
19a8d2	2
63726a	2
dae9b9	2
2bb403	2

182	901
183	903
184	905
185	907
186	908
187	911
188	912
189	915
190	925
191	935
192	936
193	938
194	950
195	952
196	962
197	966
198	975
199	990
200	991
201	996
202	998
203	1003
204	1004
205	1006
206	1010
207	1011
208	1014
209	1016
210	1020
211	1021
212	1030
213	1033
214	1034
215	1035
216	1041
217	1043
218	1045
219	1047
220	1051
221	1056
222	1057
223	1060
224	1065
225	1071
226	1073
227	1075

228	1078
229	1081
230	1083
231	1089
232	1091
233	1094
234	1095
235	1099
236	1108
237	1123
238	1125
239	1128
240	1129
241	1138
242	1152
243	1153
244	1155
245	1156
246	1159
247	1160
248	1175
249	1180
250	1193
251	1194
252	1196
253	1198
254	1200
255	1203
256	1204
257	1206
258	1212
259	1225
260	1231
261	1232
262	1235
263	1240
264	1242
265	1245
266	1247
267	1248
268	1253
269	1262
270	1264
271	1266
272	1268
273	1269

274	1271
275	1272
276	1284
277	1290
278	1293
279	1300
280	1305
281	1308
282	1309
283	1311
284	1321
285	1331
286	1333
287	1340
288	1341
289	1343
290	1345
291	1354
292	1361
293	1362
294	1365
295	1367
296	1381
297	1387
298	1393
299	1398
300	1401
301	1404
302	1417
303	1427
304	1429
305	1431
306	1432
307	1433
308	1438
309	1440
310	1446
311	1447
312	1449
313	1452
314	1454
315	1456
316	1466
317	1469
318	1471
319	1473

320	1474
321	1482
322	1488
323	1489
324	1493
325	1499
326	1523
327	1539
328	1541
329	1552
330	1552
331	1553
332	1561
333	1565
334	1571
335	1578
336	1586
337	1586
338	1588
339	1590
340	1601
341	1603
342	1607
343	1610
344	1619
345	1626
346	1633
347	1637
348	1638
349	1640
350	1646
351	1657
352	1660
353	1662
354	1666
355	1668
356	1671
357	1676
358	1676
359	1681
360	1713
361	1719
362	1742
363	1744
364	1746
365	1748

366	1755
367	1759
368	1769
369	1770
370	1774
371	1777
372	1786
373	1790
374	1798
375	1808
376	1809
377	1813
378	1815
379	1824
380	1833
381	1844
382	1845
383	1871
384	1878
385	1880
386	1883
387	1884
388	1894
389	1904
390	1904
391	1908
392	1909
393	1910
394	1911
395	1922
396	1927
397	1930
398	1938
399	1940
400	1943
401	1944
402	1947
403	1948
404	1949
405	1950
406	1952
407	1958
408	1961
409	1965
410	1979
411	1982

412	1983
413	1987
414	1988
415	1990
416	1998
417	1998
418	2000
419	2012
420	2012
421	2012
422	2017
423	2024
424	2029
425	2041
426	2043
427	2045
428	2051
429	2056
430	2056
431	2057
432	2060
433	2069
434	2076
435	2084
436	2086
437	2088
438	2103
439	2108
440	2109
441	2111
442	2115
443	2117
444	2119
445	2121
446	2121
447	2123
448	2170
449	2170
450	2175
451	2177
452	2180
453	2182
454	2184
455	2191
456	2194
457	2195

458	2198
459	2201
460	2204
461	2205
462	2212
463	2215
464	2217
465	2218
466	2219
467	2220
468	2234
469	2234
470	2234
471	2242
472	2243
473	2246
474	2248
475	2251
476	2253
477	2254
478	2256
479	2263
480	2266
481	2272
482	2281
483	2285
484	2292
485	2300
486	2301
487	2313
488	2314
489	2315
490	2316
491	2319
492	2348
493	2351
494	2352
495	2355
496	2358
497	2391
498	2394
499	2396
500	2403
501	2403
502	2406
503	2406

504	2408
505	2415
506	2418
507	2424
508	2425
509	2430
510	2443
511	2456
512	2457
513	2465
514	2472
515	2494
516	2497
517	2504
518	2506
519	2507
520	2509
521	2510
522	2511
523	2512
524	2513
525	2514
526	2515
527	2521
528	2525
529	2530
530	2535
531	2538
532	2539
533	2547
534	2554
535	2562
536	2580
537	2587
538	2595
539	2598
540	2601
541	2601
542	2605
543	2606
544	2608
545	2612
546	2617
547	2619
548	2625
549	2627

550	2628
551	2636
552	2638
553	2645
554	2647
555	2649
556	2650
557	2651
558	2652
559	2654
560	2659
561	2662
562	2664
563	2678
564	2679
565	2683
566	2685
567	2700
568	2709
569	2712
570	2714
571	2723
572	2727
573	2731
574	2735
575	2739
576	2761
577	2763
578	2764
579	2771
580	2773
581	2776
582	2786
583	2792
584	2793
585	2795
586	2796
587	2800
588	2808
589	2809
590	2810
591	2814
592	2814
593	2818
594	2826
595	2831

596	2833
597	2836
598	2840
599	2843
600	2844
601	2852
602	2855
603	2857
604	2858
605	2859
606	2861
607	2865
608	2868
609	2878
610	2888
611	2891
612	2895
613	2896
614	2898
615	2902
616	2905
617	2907
618	2908
619	2911
620	2911
621	2916
622	2916
623	2919
624	2930
625	2931
626	2949
627	2956
628	2964
629	2964
630	2967
631	2972
632	2976
633	2979
634	2983
635	2986
636	2988
637	3002
638	3008
639	3009
640	3012
641	3030

642	3031
643	3034
644	3036
645	3043
646	3045
647	3053
648	3054
649	3062
650	3063
651	3068
652	3071
653	3079
654	3086
655	3104
656	3106
657	3111
658	3114
659	3120
660	3125
661	3137
662	3139
663	3140
664	3144
665	3145
666	3146
667	3147
668	3155
669	3155
670	3156
671	3156
672	3163
673	3165
674	3174
675	3178
676	3189
677	3192
678	3193
679	3197
680	3200
681	3203
682	3205
683	3206
684	3208
685	3210
686	3215
687	3217

688	3219
689	3221
690	3223
691	3234
692	3238
693	3252
694	3256
695	3259
696	3260
697	3262
698	3268
699	3273
700	3280
701	3285
702	3286
703	3293
704	3297
705	3311
706	3320
707	3321
708	3328
709	3329
710	3333
711	3339
712	3342
713	3353
714	3372
715	3372
716	3374
717	3375
718	3375
719	3379
720	3386
721	3388
722	3397
723	3404
724	3411
725	3413
726	3419
727	3422
728	3426
729	3428
730	3430
731	3434
732	3436
733	3439

734	3440
735	3440
736	3453
737	3459
738	3460
739	3466
740	3470
741	3475
742	3484
743	3488
744	3491
745	3493
746	3496
747	3496
748	3497
749	3500
750	3503
751	3505
752	3508
753	3509
754	3515
755	3518
756	3525
757	3533
758	3536
759	3538
760	3545
761	3550
762	3555
763	3556
764	3560
765	3573
766	3575
767	3587