Bicycle Mode Activity Detection with Bluetooth Low Energy Beacons*

Paulo Ferreira
Department of Informatics
University of Oslo
Oslo, Norway
paulofe@ifi.uio.no

Andriy Zabolotny and João Barreto
Department of Computer Science and Engineering
IST/UL/INESC ID
Lisboa, Portugal
andriyzabolotnyy@tecnico.ulisboa.pt and joao.barreto@tecnico.ulisboa.pt

***Abstract*—In a growing number of cities, cycling is being seriously considered to help solving traffic congestion, parking, etc. It is also a cleaner and healthier mean of urban transportation. However, changing users behavior (e.g., using a bicycle instead of a car) is not simple. Thus, to promote and motivate cycling we propose Biklio, a cycling rewarding system that, based on the use of a smartphone, detects when a user starts cycling and makes her/him eligible for rewards. This solution uses a smartphone application that includes a bicycle usage detection component. This component is both highly accurate and cheap, while respecting other requirements, and is based on the use of a Bluetooth Low Energy (BLE) sensor, installed on each bicycle, which is detected by smartphones. The system is implemented and running, and the results obtained are very encouraging.**

***Index Terms*—smartphone, BLE sensor, mobile, bicycle.**

I. Introduction

In urban environments, cycling may help to solve traffic delays or interruptions, parking, etc., being at the same time a cleaner and healthier means of urban transportation. Thus, several approaches are being taken to increase cycling in urban areas [5]. One such approach, Biklio (www.biklio.com), takes advantage of the availability of smartphones and it aims at motivating users to cycle by rewarding such an activity. In fact, current smartphones have many sensors (e.g., GPS, accelerometer, gyroscope, etc.) and exist in large numbers (in most developed countries, smartphone penetration reaches more than 80% of the population [1]) making them an interesting and relevant platform for collecting users data [2], [3], [7], [9].

The Biklio system includes a smartphone app (for short, BiklioApp), a Bluetooth Low Energy (BLE) sensor, a web server, and a database. The BiklioApp runs in both Android and iOS smartphones; it detects when a user starts cycling and makes her/him eligible for rewards that can be claimed at the associated shops (inserted by their owners into the Biklio system using a browser that interacts with the web server mentioned above). When a user gets close to or enters into such a shop, the BiklioApp detects it and allows the user to easily claim the benefit. The Biklio database stores all the data related to the associated shops (e.g., their name, location, logo, reward given to cyclists, etc.) as well as Biklio users (e.g., email, trajectories, etc.)

The goal of this article is to describe the design, implementation and evaluation of the Biklio module aimed at detecting user cycling activity with a high confidence level (ideally 100%), within a limited amount of time (i.e., less than 30 sec. after having stopped cycling), while being aware of the resource-constrained characteristics of smartphones, energy-wise in particular. Note that Biklio is a system much bigger than the cycling activity detection module. However, in this paper we focus on the BiklioApp and, particularly, on the cycling activity detection solution.

Solutions aiming at detecting human activity recognition (HAR), in particular the transports used, currently use smartphones to collect data from users. The activity data collected is processed either remotely (i.e., on a server) or locally (i.e., in the smartphone). On one hand, **remote** data analysis means that data is gathered in some server (e.g., in the cloud) where a machine-learning algorithm is executed, and a conclusion is reached with some confidence level (which currently is always less than 100%); obviously, regarding responsiveness, such a conclusion (the transport mode) is reached long after the activity takes place. In addition, such a solution requires a smartphone to connect to the network and receive the result; this obviously adds more time to the detection (when compared to a detection on the smartphone itself) or may not even be possible.

On the other hand, **local** data analysis means that data is collected and, even if it is processed in some server (e.g., training a machine learning algorithm that then generates a classifier), the detection itself (e.g., done by a classifier) is done in the smartphone. The algorithm used (e.g., the classifier mentioned above) must be adapted to a resource-constrained device (when compared to a cloud server) and provides a conclusion regarding the activity being performed (the transport mode in this case) almost instantaneously with some level of confidence (which currently is always less than 100%).

Most existing solutions (both remote and local) do not consider the use of a bicycle as a transport mode. However,

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID), FCT scholarships SFRH/BD/ 103745/2014, SFRH/BSAB/135197/2017, and project PTDC/EEI-COM/30644/2017.

978-1-7281-2522-0/19/31.00 ©2019*IEEE*

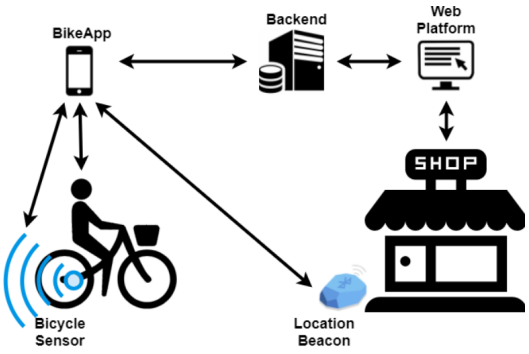

Fig. 1. High level global view of Biklio.

Fig. 2. Biklio bicycle-mounted sensor.

bicycles are more and more prevalent in urban environments, as urban planners and citizens agree about the key role of the bicycle towards more efficient, livable and sustainable cities [8].

To guarantee correct rewarding, i.e. giving rewards to users who have effectively ridden a bicycle for a pre-determined time and/or distance, the BiklioApp needs to detect the cycling activity, and determine a user's location, to ensure that she/he is close to a given shop. User's location is out of scope of this article and several solutions exist that can be used [4], [6], [10]; currently, we use either GPS (available in most smartphones) or Bluetooth Low Energy sensors for proximity detection (e.g, from Estimote), but others could be used.

To detect cycling activity, the BiklioApp uses a BLE bicycle-mounted sensor. This sensor pairs with a smartphone so that, when both are close to each other, the smartphone detects the sensor; thus, the BiklioApp running on the smartphone detects the bicycle's proximity. In addition, the BiklioApp running on the smartphone, while the bicycle is being used, receives data from the BLE sensor that allows it to calculate the bicycle's speed.

The remainder of this paper is organized as follows. Section II describes the architecture of the BiklioApp and the most relevant aspects of its implementation. Section III presents some evaluation results. The paper ends with a section on conclusions.

II. Biklio Architecture and Implementation

In Biklio, the web platform (see Figure 1) allows shop owners to register their shops and become part of the rewarding program. During the registration, basic shop information is requested, such as name, type, address and logo. The shop owner also specifies the reward it is offered to cycling customers (e.g., a 5% or 10% discount, free coffee, etc.).

The backend plays an intermediary role since it handles requests both from the web platform and from the mobile application. It stores in a database all the application relevant data, such as users, registered shops information, etc. The BiklioApp opportunistically synchronizes with the backend (i.e., when there is an active Internet connection). For example, if a user runs the BiklioApp in off-line mode and her/his route is tracked, the registered route trajectory (set of of GPS coordinates) is only uploaded to the backend when network connection becomes available.

The BiklioApp provides information about nearby shops that joined the rewarding program, which are shown in the app main screen, that users see upon opening the BiklioApp; it presents a map with the nearby shops (with respect to a user's location) with the corresponding reward description and distance. When opening the "Map" tab, the application presents a map with the nearby shops. Then, specific shop information can be found in the screen.

The BiklioApp has the notion of eligibility for each type of reward; this means that a reward can be claimed (at some shop) when a user becomes eligible for the corresponding reward. This happens once the BiklioApp detects the cycling activity during a certain period of time. Currently, this is a very short time (1.5 minutes) given that we are interested on encouraging people to use the application.

When a user is close to or enters one of the associated shops, and is eligible for a reward, the BiklioApp notifies her/him that there is an available benefit to claim. Then, upon opening the application, a benefit claim screen pops up that contains the information about the detected shop and the reward description. To claim the benefit, all the user needs to do is press the "Claim Your Benefit" button and show the corresponding screen to the shop cashier, who will give the user the benefit. The claimed benefit screen shown in the BiklioApp contains the shop logo, the reward description, and the moment at which the benefit was claimed.

A. Cycling Detection

Biklio's cycling detection uses a BLE bicycle sensor that is attached to a bicycle frame and contains two components: one magnet attached to the pedal, and another attached to the rear wheel spoke (Fig. 2 shows the sensor we currently use).

So, the first step requires a user to establish a pairwise relation between the bicycle-mounted sensor and her/his smartphone. Note this step is done only once, i.e., when the bicycle-mounted sensor is used for the first time (e.g., when the sensor is installed), and is similar to the pairing done between a smartphone and an audio car system. Thus, we believe this is a simple step for any user.

(a)	wheelRevolutions = currentWheelRevolutions - previousWheelRevolutions
(b)	crankRevolutions = currentCranksRevolutions - previousCrankRevolutions
(c)	wheelEventElapsedTime = currentLastWheelEventTime - previousLastWheelEventTime
(d)	crankEventElapsedTime = currentLastCrankEventTime - previousLastCrankEventTime
(e)	instantaneousSpeed = (wheelRevolutions * CIRCUMFERENCE) / (wheelEventElapsedTime*1024*3.6)
(f)	instantaneousCadence = (crankRevolutions/crankEventElapsedTime)*1024*60

TABLE I
FORMULAS USED IN BIKLIO.

The BLE Manager module in BiklioApp manages the connection and the communication with the BLE bicycle-mounted sensor. It scans for nearby BLE devices, establishes a connection with a specific device, and receives device updated data. This data is sent by the sensor once it is activated through pedal or wheel revolution. So, after a connection is established, the BLE Manager starts receiving sensor data, and routes it to the Cycling Detector module. The latter has no knowledge about the sensors and simply has the function of interpreting the sensor data to determine the cycling state.

Note that the connection between the BiklioApp and a bicycle-mounted sensor just means that a user is nearby a bicycle, and not necessarily cycling. To detect the cycling activity, the BiklioApp uses the instantaneous cadence and speed values that are calculated on the basis of the data received from the sensor (as described later).

The sensor data mentioned above contains four cumulative values: the total counts of crank and wheel revolutions, and the times the last crank and wheel revolutions occurred. We designate them Crank Revolutions, Wheel Revolutions, Last Crank Event Time, and Last Wheel Event Time, respectively. Each one of the counters starts with the value 0, i.e. when the sensor is activated for the first time after being bought, and is always incremented.

As previously mentioned, there is a magnet attached to the pedal and another to the rear wheel. When the user cycles, each pedal revolution makes the magnet pass near the sensor, thus incrementing the Crank Revolutions counter by 1. Similarly, the sensor increments the Wheel Revolutions counter with each wheel revolution. The Last Crank Event Time and Last Wheel Event Time are incremented with the elapsed time in units of 1/1024 of a second, every time there is a crank and a wheel revolution, respectively. For example, if a revolution takes 2 seconds, the value is incremented by 2048.

The bicycle-mounted sensor periodically sends the values of each one of its four cumulative counters within a single data packet. Using simple math, the Cycling Detector module determines the instantaneous wheel and crank revolution speeds, as described next. Let us consider two consecutive received data packets, and use ”previous” and ”current” to identify the counters values contained within each packet. The expressions in Table I (formulas a-d) show how to determine the number of revolutions and the elapsed time, between two sequential received packets, by simply subtracting their counters.

To determine the instantaneous wheel speed, the BiklioApp multiplies the previously calculated *wheelRevolutions* by the wheel circumference (in meters), which gives the traveled distance, and then divide it by *wheelEventElapsedTime* (see Table I-e). Given that the time is in units of 1/1024 of a second, and the traveled distance is in meters, the BiklioApp multiplies by 1024 and 3.6 to get the instantaneous speed in km/h. To determine the instantaneous cadence value (see Table I-f) the BiklioApp divides *crankRevolutions* by the *crankEventElapsedTime* and finally, multiplies by 1024 and 60 to get the cadence in RPM.

Currently, the BiklioApp uses a value of 210 cm for a wheel circumference (formula (e) shown in Table I). This value corresponds to a typical bicycle with a 26-inch wheel size. If a different bicycle is used, the wheel circumference value is easily changed in the BiklioApp settings interface.

The Cycling Detector module considers that a user is cycling when the instantaneous speed and cadence values exceed 11 km/h and 30 RPM during a determined period of time. Such values were determined as corresponding to cycling at a slow speed but faster than walking.

Another circumstance that must be considered is when a cyclist is effectively using her/his bicycle and stops for a short moment (e.g., in a red traffic light) and then continues. This should not be interpreted by the BiklioApp as the user finishing her/his bicycle trip. Thus, these two scenarios (temporary intermediate stop, and final stop) are considered as follows.

The Cycling Detector detects that a user is no longer cycling (having reached her/his final destination) when the corresponding module does not receive updated sensor data for a determined period of time. This period of time must be such that it is able to consider the case in which a user just stops cycling for a few seconds (e.g., in a red traffic light). If such a detection was done erroneously, that would result in finishing the cycling detection when the user did not end the trip effectively. Through experiments, we concluded that a value of 30 seconds for such a period of time is a good fit.

Note that with respect to tracking the route of a cyclist (e.g., when the reward depends on the distance cycled) there is no significant relevance on the chosen timeout value. Even if the timeout elapses, the tracking for that particular route finishes but it will instantly start again as soon as the user resumes cycling. Thus, the total tracked distance will still be the same, not affecting the eligibility for the cycled-distance rewards, in particular.

Another relevant aspect is related to the fact that, sometimes, a user may abruptly push her/his bicycle resulting in high instantaneous values sent by the sensor; this clearly does not correspond to a user cycling as it is a spurious event. Thus, to deal with such a situation, the BiklioApp uses a time period

within which the instantaneous cadence and speed values can be higher than the thresholds. Our experiments have shown that a period of 5 seconds is adequate.

B. Implementation

The BiklioApp is developed in a cross-platform fashion using Xamarin (https://www.xamarin.com/) as it allows developers to write, using the C# programming language, both Android and iOS apps, and share code across multiple platforms. More specific (low-level) code requires either Swift or Java programming (for iOS and Android, respectively).

For cycling detection, the BiklioApp uses a bicycle attachable BLE sensor from BTWIN. Figure 2 shows the sensor mounted on a bicycle. The application was tested on an iPhone 5s with iOS 10.3.2; the evaluation results presented in the next section use the above mentioned equipment.

III. Evaluation

We evaluated the following aspects of the cycling detection solution in the BiklioApp: i) accuracy of the detection itself (i.e., the capability of the system to detect that a user is effectively cycling with neither false positives nor false negatives), ii) the time it takes to connect the BiklioApp, running on the smartphone, to the bicycle-mounted sensor, iii) the time that the BiklioApp takes to detect the cycling activity, iv) the battery consumption induced by the BiklioApp on the smartphone due to the cycling activity detection, and v) the usability of the BiklioApp. For lack of space, we only present the accuracy of the detection and a summary of the battery consumption. Note that the other results do respect the requirements.

A. Cycling Detection Accuracy

The tests done (more than 10 trips) have shown a result that did not change with the number of users and trips: 100% correct results. Thus, each time a user started cycling, the BiklioApp (either running in the background or in the foreground) managed to detect cycling activity with no errors. Note that 100% is not a surprising result; as a matter fact, this is equivalent to the establishment of a pair-wise relation between a Bluetooth device and the audio system of a car, and the corresponding usage to make and answer phone calls.

B. Battery Consumption

First, we observe the power consumption of a smartphone in standby (i.e., without running any application); this is the reference value. Then, we determine the power consumption of the BiklioApp when the smartphone is idle, and when a user is cycling.

We left the smartphone overnight registering the battery level changes into a logfile for the two scenarios mentioned above: i) not running any application, and ii) running Biklio when the smartphone is idle. In both scenarios, we consider the battery consumption over a period of 12 hours. The results for battery consumption/hour obtained are the following: i) no application: 0.40%, and ii) the BiklioApp (with BLE sensor): 0.74%. The results show that when the smartphone is idle (no application is running), its shows less power consumption than with the BiklioApp (0.40%). However, we can also see that the continuous background BLE scanning does not introduce a big power consumption overhead, as the consumption just increases by 0.34% (up to 0.74%).

We also developed a GPS based solution to compare its battery consumption with Biklio sensor based approach. Under the same circumstances, the battery consumption for this case is 2.75% which, as expected, it is clearly much higher than the solution used in the BiklioApp with the BLE sensor. In addition, the accuracy of the GPS solution is less than 100% as obtained with the BiklioApp (with the bicycle-mounted sensor).

IV. Conclusions

This paper presents Biklio with a focus on the cycling detection performed by the BiklioApp. Such detection has very strict requirements which are all respected by the current system. It is based on a Bluetooth sensor attached to each bicycle which is read by any BLE smartphone capable of running the BiklioApp. The results obtained show that the solution provides high accuracy on cycling activity detection, it is easy to deploy, it requires low or no maintenance, it is user-friendly and minimally intrusive, it has low battery consumption, it is responsive, and it is cheap as it uses a off-the-shelf BLE sensor.

References

[1] Global mobile consumer trends, 2^{nd} edition. Technical report, 2017.

[2] O. C. Ann and L. B. Theng. Human activity recognition: A review. In *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, pages 389–393, Nov 2014.

[3] M. A. Ayu, S. A. Ismail, A. F. A. Matin, and T. Mantoro. A comparison study of classifier algorithms for mobile-phone's accelerometer based activity recognition. *Procedia Engineering*, 41:224 – 229, 2012. International Symposium on Robotics and Intelligent Sensors (IRIS 2012).

[4] S. Bricka and C. R. Bhat. A comparative analysis of GPS-based and travel survey-based data. *Transportation Research Record*, (1972):9–20, 2006.

[5] J. Hrnčíř, P. Žilecký, Q. Song, and M. Jakob. Practical multicriteria urban bicycle routing. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):493–504, March 2017.

[6] A. Lari. Automated Transportation Mode Detection Using Smart Phone Applications via Machine Learning: Case Study Mega City of Tehran. *Transportation Research Board 94th Annual Meeting*, 6147, 2015.

[7] M. Nikolic and M. Bierlaire. Review of transportation mode detection approaches based on smartphone data. 2017.

[8] M. S. Rusman., N. M. P. Toyong, K. N. M. Yusuff, and M. F. Kamaruzaman. Role of bicycle as a sustainable transportation in universiti teknologi mara. In *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, pages 780–783, April 2013.

[9] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085, 2015.

[10] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. Understanding mobility based on GPS data. *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*, (49):312, 2008.