KETCube – the Universal Prototyping IoT Platform

Jan Bělohoubek, Jiří Čengery, Jaroslav Freisleben, Petr Kašpar and Aleš Hamáček
Faculty of Electrical Engineering
University of West Bohemia in Pilsen
Pilsen, Czech Republic
{belohoub, cengery5, jafre, petrx, hamacek}@ket.zcu.cz

***Abstract*—The IoT (Internet of Things) devices continue to penetrate into new areas of our daily lives as well as industry. The evolution of IoT devices comes with the necessity of operation in heterogeneous environments. This evolution brings new challenges in the areas of R&D and education. We identify important features beneficial for R&D engineers as well as for educationalists and students and we propose a novel open platform for rapid development of IoT nodes. This platform is easy to employ in the educational process at the same time.**

I. Motivation

The predictions show, that within the next decade, over a trillion new sensors will be deployed per year [1]. Additionally, the so called IoT (Internet of Things) devices continuously appear in new contexts and new application areas. The growing IoT sector powers the so called *Fourth Industrial Revolution* or *Smart anything/everything* initiatives like *Smart Cities* and *Smart Farming* [1], [2].

The continuous and fast movement in the application area stimulates the progress in many fields – IoT LPWAN (Low Power Wide Area Network) standardization efforts [3], infrastructure efforts like LoRaWAN, Sigfox, NB-IoT and many more [3], [4]. Development of novel sensors [1] and LPWAN nodes or the continuously growing area of so-called "big data" processing [5] and control [6].

As the IoT field in general is very heterogeneous [3], developers of physical devices enjoy not only the advantages of the heterogeneity, but they also face many challenges. Although the heterogeneity is positive for developers' freedom to choose the best solution for a particular problem, it represents a great challenge when device is deployed. One of the significant challenges for IoT nodes is the (in)ability to gain the profit coming from overlapping networks based on different communication standards [4]. It is an serious challenge especially for LPWAN nodes, targeted on multi-region usage independently of the LPWAN network with the best coverage in a particular region.

Given the heterogeneity, it is simple to imagine, that a single IoT device project moves through a support of multiple networks even during different development stages: point-to-point wireless links can be used in a technology demonstrator phase, another network can be used for prototyping and validation and additional networks should be supported when deployed. This enforces the repetition of a validation phase and makes the overall process much more complicated and time-consuming.

Another challenge coming from the developing IoT area is connected with the technical education [7]. Educationalists all-around the world face the problem how to introduce students to IoT world without missing any important technology while providing detailed technical insight at the same time.

In many classes all-around the world, the Arduino® platform is used by educationalists as the main hardware platform for low power and low performance nodes [7]. As Arduino® is the first choice for beginners coming with number of advantages following the diversity of IoT world, it also has significant disadvantages. The important disadvantage from the educational point of view is that the simplicity and the number of simple HowTos and libraries (of varying quality) do not stimulate the student's personal knowledge progress and insight.

According to our experience, when Arduino® is used, students are able to design an operational device very fast. Unfortunately, they often don't understand what the device really does, how to optimize the device's power consumption, what is the difference between I^2C and SPI etc. Here, the simplicity of Arduino® makes it harder to get insight. Additionally, the usage of Arduino® does not stimulate the ability to understand the industry-level style of documentation – datasheets or application notes. Students often tend to follow short HowTos and step-by-step guides of varying quality.

On the other hand, if the class is concentrated on data processing only, IoT nodes can be seen as black-boxes producing data. Similarly, if students develop sensor elements, IoT nodes would be used e.g. as analog-to-digital converters equipped with a network interface. Based on these use cases, one would appreciate not to waste time with Arduino® programming and debugging – a platform even simpler than Arduino® (from a user's perspective), if available, would do the job more efficiently.

As described above, there is a room for a platform fulfilling the following criteria: (i) support for multiple LPWANs, (ii) unified and modular approach to sensor and actuator elements enabling rapid prototyping, (iii) to be validated once and deployed to different environments, (iv) an open source approach enabling the most effective educational usage, (v) easy to use at a user-level (configuration tools and user-level documentation) and at a developer-level (documentation, open-source), (vi) high-quality, an industrial-level documentation enabling both: educational and commercial use.

The rest of the paper is organized as follows: in Section II, we describe the novel educational and prototyping platform

for IoT developed at the University of West Bohemia in Pilsen fulfilling the mentioned criteria. In Section III, several use cases are described. In Section V, we present the current plans and future work. The paper is enclosed by Conclusions in Section IV.

II. KETCube Platform

Based on the IoT design and educational experience of our team, we decided to release the new prototyping and educational platform supporting our R&D process as well as our educational activities under the non-restrictive University of Illinois/NCSA Open Source License [8]. We call our platform KETCube. The name of the KETCube platform consists of the abbreviated name of the institution of its origin – Department of Technologies and Measurement (KET), University of West Bohemia in Pilsen – and the shape of the basic sensor node – a cube.

The KETCube platform itself represents the *"point of integration"* for wide range of heterogeneous software and hardware modules. The platform is not intended to be a new *"field-breaking"* tool coming with novel approaches. Instead the KETCube synthesizes many state-of-the-art approaches to provide powerful IoT platform. The added value of KETCube is that it provides an unique, well documented, simple to employ and simple to extend ecosystem for integration of different modules. The emphasis is on the usage of industry-level (but simple enough) approaches enabling efficient usage of the KETCube platform as an educational and prototyping tool in the same time.

The platform itself consists of three parts: Hardware (asembled Printed Circuit Boards – PCBs and the Box), Firmware and Documentation. Each part is designed in such a way, that it can be used independently of others and it is compatible with a number of the 3rd party parts while using all KETCube platform parts together is advantageous. E.g. the Firmware can be used with the 3rd party compatible boards[1,2]; PCBs can be used with the 3rd party firmware[3,4] and the documentation naturally as a reference for related projects.

As every platform part can be used separately, from the project management point of view, multiple repositories are used: a single repository is used for firmware[5], another repository for documentation[6] and for the main board[7] resources. The additional board design files and documentation reside in separate repositories. Even though the project structure is flat, the general platform documents keep all parts together and explain connections and synergies between platform components – thus the best starting point is the documentation repository serving as a project home-page.

In order to increase the educational potential of the KETCube platform, open or at least free-of-charge industry-level tools are always preferred to perform any project-related task. The quality and width of available documentation has high priority while preserving documentation portioning to user-level documents (unified and well defined format) and task-specific developer-level documents (varying format – depends on the task type).

A. Hardware

The important feature introduced on the KETCube platform is a semi-standardized KETCube socket – see Figure 1. The KETCube socket allows to use a wide range of ready-to-use boards [9], as the socket is compatible with a widely used [9] mikroBUS™ socket (a part of header H4 and header H5), thus mikroBUS™-compatible boards can be used with the KETCube.

The KETCube socket is a superset of the mikroBUS™ with 4 additional configurable (solder jumpers) IO pins (IO1 – IO4) and the increased space between headers (headers H4 and H8; from 22.86mm = 0.9" at the mikroBUS™ to 29.21mm = 1.15"), bringing more space for e.g. inter-header placement of batteries. Just two (H4 and H8) or even all (H4, H5 and H8) headers can be placed on a KETCube-compatible board.

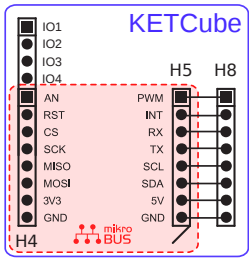

Fig. 1. KETCube socket and mikroBUS™ – a subset of the KETCube socket

The heart of the platform is the 32x32mm *main board* equipped with the KETCube socket and well-established ARM-based SMT32L0 MCU [10] together with the Semtech's *SX1276* radio [11] in a single package deployed by Murata as *Type ABZ LoRa Module* [12]. Additionally, the Ti's RHT (Relative Humidity and Temperature) sensor *HDC1080* [13] allows to use the *main board* itself as a simple environmental sensor.

As the KETCube socket headers H4 and H8 are placed at the board edges, the recommended pass-through headers allow almost infinite stacking of KETCube-compatible boards. Additionally, one mikroBUS™-compatible board can be stacked at the top of the KETCube, as the *main board* is equipped with headers (H4, H5 and H8) corresponding to the mikroBUS™ layout. The only inconvenience is, that if the KETCube socket is occupied by an mikroBUS™ board, stacking of additional boards at the top is disabled, while stacking at the bottom is still possible.

Depending on the user's needs, a bulk antenna, SMA or ULF connectors can be assembled. 10 solder jumpers increase the adaptability of the board: besides they enable pin assignment options (IO1 – IO4), power source selection, etc. [14].

The MCU can be programmed through the on-board SWD (Serial Wire Debug) socket. The *main board* can be powered through micro USB (5V) or a 3V3 socket pin.

[1] Grasshopper LoRa Development Board (https://www.tindie.com/products/TleraCorp/grasshopper-lora-development-board/)
[2] B-L072Z-LRWAN1 (https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html)
[3] https://github.com/GrumpyOldPizza/ArduinoCore-stm32l0
[4] https://github.com/kriswiner/CMWX1ZZABZ
[5] https://github.com/SmartCAMPUSZCU/KETCube-fw
[6] https://github.com/SmartCAMPUSZCU/KETCube-docs
[7] https://github.com/SmartCAMPUSZCU/KETCube-mainBoard

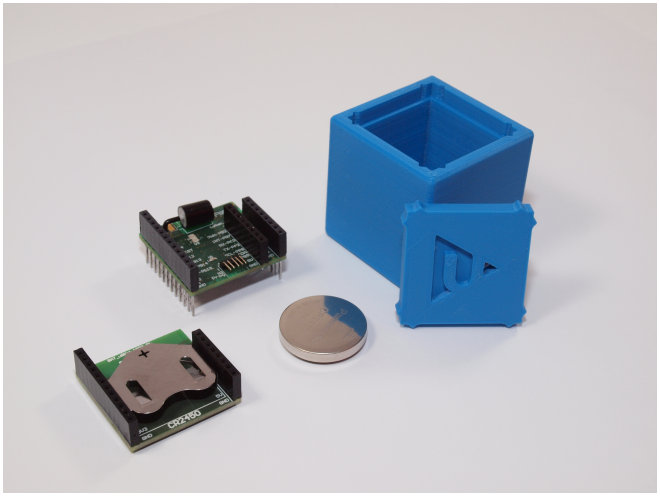

Fig. 2. KETCube platform hardware: the *main board* with a bulk antenna, box and battery board with CR-2450

The second PCB included in the current project release is the *battery board* for CR-2450 allowing the KETCube battery operation. The CR-2450 is recommended for evaluation only.

A typical KETCube based system contains a *main board*, a *battery board* (with a KETCube socket and battery) and a sensor board connected to the KETCube or mikroBUS™socket. Up to three KETCube boards can be stacked together and placed into a pre-designed cube-box – see Figure 2. If more than three boards are used or the physical layout of boards stacked on the *main board* overlaps the physical size of the *main board*, a custom box must be used.

For detailed description, see KETCube Datasheet [14].

B. Firmware

The KETCube firmware is divided into two logical parts: *Core* and *Modules*. The structure of the firmware stack is in Figure 3. The hierarchical structure of the stack reflects the mission of the platform while keeping it as clear and simple as possible.

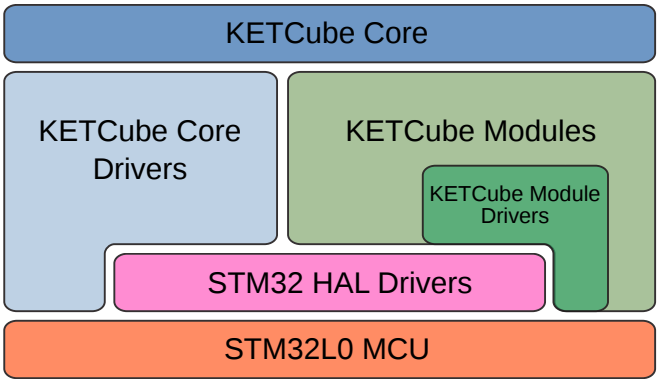

Fig. 3. KETCube firmware stack

The KETCube Core is the principal part of the KETCube platform firmware. It is not closely related to the underlying hardware and it is designed to be easily ported to any MCU platform if necessary. The KETCube Core is not an operating system.

The hardware-related part required by the KETCube Core is denoted as the *KETCube Core Drivers* and it contains only drivers employed exclusively by the KETCube Core: the EEPROM driver enabling the non-volatile configuration storage and the serial line driver required for the KETCube command line interface.

On the other hand, *KETCube module* implements a particular functionality, e.g. LoRaWAN, an on-chip ADC sensor or a specific I2C slave support. The module can closely interact with hardware by using STM32 HAL drivers or even lower-level drivers [10]. The decision fully depends on a module maintainer, that should consider the module nature, code readability, decomposition and maintainability [15]. Usage of existing *Module Driver(s)* is recommended if possible. Writing a new driver is recommended, when there is a potential of re-usability of an introduced driver.

Modules are further divided depending on their purpose to: *Sensing Modules*, *Communication Modules* and *Actuating Modules*.

The KETCube firmware operates according to the diagram in Figure 4. Most of the useful work is performed by periodic tasks (one task per module is executed with the *system period*) and in the meantime, the KETCube stays in a low-power mode (interrupts are handled). With the frequency given by the system period, all sensing modules produce data: these are passed to (all enabled) communication module(s) and transmitted. Data (commands) received by communication module(s) are passed to respective actuating (or configuration) modules (*inter-module messages*) and processed (executed).

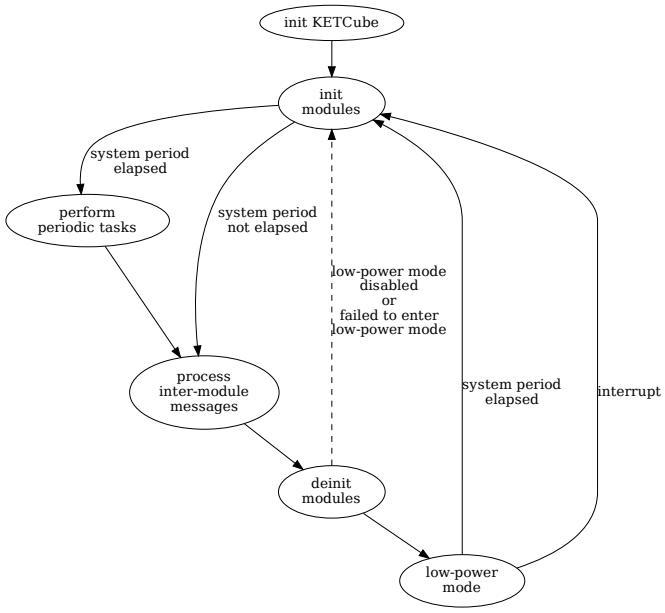

Fig. 4. Simplified KETCube firmware operation diagram

The responsibility of each module is to translate data formats between the inter-module KETCube format and the intra-

module formats, which can be specific to particular technology implemented by the module itself (e.g. LoRaWAN stack).

The approach used for handling the data is as straightforward and as simple as possible. Data transmitted from the KETCube are sequenced in the order given by enabled sensing modules (more modules enabled → longer data frames transmitted). To be able to decode the data received from the KETCube properly, one must only know which modules were enabled on the originating KETCube and the format of data produced by used modules. For multi-byte variables, BigEndian is preferred.

The data received by the KETCube must (implicitly or explicitly) contain the index of a destination module. This can vary for different communication modules – e.g. port numbers are used by LoRaWAN module – the communication module's responsibility is to translate a module-specific format to the KETCube core representation.

As the KETCube firmware is highly modular and configurable at both – source-code and binary – levels, any module can be enabled/disabled individually by using preprocessor definitions at compile time (module linked or not) or by using the KETCube command line interface at run-time (module On/Off). The whole firmware uses exclusively static data structures. Included modules are statically registered and the core uses a defined callback interface to deal with them.

As the introduction of the KETCube platform was driven also by the educational potential of such work, the code readability and maintainability is an absolute priority [15] together with availability of used tools. Multiple open and commercial IDEs including Keil μVision, Atollic TrueSTUDIO or SW4STM32 can be used to work with the KETCube project. Naming conventions [16] are defined and the *coding style* is defined in terms of well established GNU/Indent tool [17].

The complete Doxygen [18] programming documentation is provided alongside the code[8].

C. Documentation

The documentation of open hardware projects in general has varying quality [19]. The KETCube platform documentation is structured in such a way to enable multi-level recognition of the platform. The structure of the documentation follows the structure used by the leading semiconductor industry manufacturers such as *Texas Instruments*, *ST Microelectronics* or *Analog Devices* to increase the educational and usability benefits of such documentation for students as well as for developers and common users.

There are several types of documents: the root document is the KETCube platform datasheet [14], followed by application notes [16] related to particular use cases and development steps. The most detailed and partially heterogeneous documentation is represented by task-specific and annotation-generated documents (firmware documentation generated by Doxygen, reports related to PCB designs, ...).

The datasheet together with application notes are highest user-level and unified-style PDF documents related to the KETCube platform, while the deep-in theoretical introduction can be included depending on the documented use case(s) (e.g. theory of measurement). These documents can be found in the documentation repository.

The documentation closely-related to a particular part of the KETCube platform – a developer-level documentation – is placed in the same repository as the documented part (e.g. Doxygen documentation of firmware is a part of the firmware repository).

The overall structure of the documentation allows the up-down study procedure: one can start from the user-level documentation represented by the KETCube datasheet [14]. Firmware developers can benefit from structured and open Doxygen-generated documentation and firmware-related application notes [16]. Hardware developers gain from released sample designs and published application notes [16]. For those, who deploy the KETCube-based solution, also the application notes will be a valuable source of information.

D. KETCube-based Project Life Cycle

The platform is intended to be used for prototyping and validation of low-performance and low-power wireless IoT devices and technology demonstrators allowing to accelerate in particular the project prototyping, validation and testing stages – see Figure 5, while enabling fast move to a production phase.

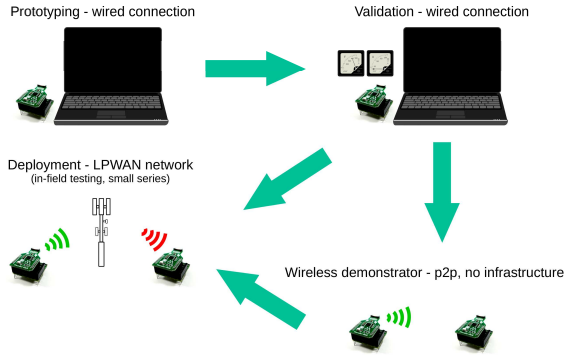

Fig. 5. Typical KETCube-based system development stages

Normally, within the KETCube-based project, at least one KETCube extension board and one KETCube firmware module are created: the extension board is stacked with the KETCube *main board* and the firmware module is integrated into the KETCube firmware.

The project development cycle may iterate several times thru stages shown in Figure 5, while reusing most of the work from previous iterations due to the hardware and software modular design.

The mass production may require a customized PCB (Printed Circuit Board) design: integration of involved boards to cut the product price.

E. Comparison with Existing Platforms

When designing an IoT device, one can start with manufacturer's reference designs or dev-kits. The significant disadvantage of many dev kits is their size. Especially for later

[8] https://smartcampuszcu.github.io/KETCube-fw

design phases, such as in-field testing, smaller devices are advantageous – this may require the test series manufacturing.

The KETCube extension-board based approach brings easy to use development platform, which is reasonably flexible, while staying reasonably small. This allows to perform most of the development stages directly with the development platform without manufacturing special test series. Even if the custom PCB will be always preferred for mass production, the original (in-development) boards can be used as final design for smaller series, as the size of KETCube-based system is reasonably small.

As the firmware development starting point, the manufacturer's firmware examples can be used. These are often good enough to design simple IoT device, but reasonable effort may be required to extend or modify them, because of missing features, lack of documentation or custom design style.

Another option is to choose an embedded operating system like *FreeRTOS*[9]. The disadvantage of such an operating system is that there is longer learning curve compared to simpler pieces of code. More complicated system may also cause significant performance and power consumption penalty when used on small low-power and low-performance MCUs. Additionally, the system complexity makes the overall system optimization more difficult. The educational usage (especially in entry-level classes) is also limited due to the system complexity.

For ARM-based platforms, ARM deploys the *Mbed OS*[10]. This platform is probably very good choice for many projects, as it has great community and is supported by one of the leaders in the semiconductor industry. After all, the KETCube fits better to many cases: as Mbed OS is open source software, you can naturally study or modify it, but this requires a lot of knowledge similarly to other embedded operating systems. The expected usage of Mbed OS involves custom Mbed tools, which allows to write just simple applications and hide all the annoying work. This results in two extremes: working with Mbed OS is extremely hard (low-level modifications) or extremely easy and simplified (applications).

The mid-complexity of the KETCube firmware allows to understand the whole firmware, while providing number of useful functions. This makes the KETCube firmware ideal for educational purposes or for projects, where firmware should be compact and easy to optimize. A disadvantage of the KETCube firmware is that it currently supports only one hardware platform, but due to limited complexity, porting to other platforms is not a significant problem.

In educational process and also for simpler commercial projects, Arduino® is often used. The Arduino® and the KETCube platforms are both designed to be as easy to modify/reuse as possible. While the Arduino® platform is universal, the KETCube is intended for LPWAN device prototyping and education only. Even though we like Arduino® as an useful tool for prototyping of almost everything, the KETCube platform is more suitable for prototyping and education in the area of LPWAN devices.

[9]https://www.freertos.org
[10]https://www.mbed.com

The following points highlight the differences with the Arduino® platform:

- the emphasis on the industry-level documentation style and quality in contrast with tutorial-based Arduino® approach,
- the priority in usage of professional (industry standard) tools including Keil μVision, Doxygen or GNU/Indent in contrast with custom and simplified tools,
- the focus on the LPWAN device prototyping and education only in contrast with universality and great diversity of Arduino®,
- the transparency of the development process is not sacrificed for simplicity. For example: Arduino® IDE implicitly hides the library code to make the overall project structure simpler. The KETCube firmware has no implicitly hidden parts, while preserving the simplicity through solid decomposition.

The following points highlight the KETCube platform advantages:

- the *ready-to-use approach*: multiple sensors can be used without writing a single line of code (KETCube user-level configuration),
- the *easy-to-extend approach*: the documented software and hardware parts and the unified design style make the KETCube platform easy to modify or extend,
- the *transparency approach*: standard approaches are used in a straightforward, clear and well documented way,
- the *mid-complexity approach*: the KETCube firmware is complex enough to simplify most annoying tasks during the IoT device development, while staying simple enough to enable fast and detailed insight,
- the *reasonable size approach*: thanks to small PCB sizes, the KETCube can be directly used for *in-field* testing or even for deployment.

III. Use Cases

In this section, we present several use cases to demonstrate the flexibility and usability of the presented KETCube platform.

A. Environmental Sensor LPWAN Node

The environmental sensing is the basic functionality represented by a number of IoT sensor nodes. The KETCube platform in the basic configuration (the *main board*, the *battery board* and the box) is a ready-to-use environmental sensor: almost no effort is needed to run the RHT sensor, when one already has the KETCube board running the KETCube firmware.

Deploying the KETCube RHT sensor requires just to set the network parameters by using the serial terminal interface (Figure 6) and configure the device in a network (e.g. LoRaWAN server configuration).

```
> enable HDC1080
> enable LoRa
> set LoRa OTAA
> set LoRa appEUI 1122334455667788
> set LoRa appKey 11223344556677881122334455667788
> reload
```

Fig. 6. KETCube serial terminal confiruration example: LoRa OTAA and HDC1080 RHT sensor

Fig. 7. KETCube as an RHT sensor fasten to a ferromagnetic surface

The KETCube can be placed in the monitored environment to gain data – see Figure 7. The permanent magnet, which is an optional part of the KETCube box can be used to fasten the KETCube to a ferromagnetic surface.

B. Object Presence Sensing Demonstrator

The KETCube has been used as the base platform for a technology demonstrator for wireless textile capacitive sensor. This case demonstrates the flexibility of the presented platform, as it is used in an unusual scenario. The KETCube is not used as an LPWAN node but two KETCubes are used to establish a point-to-point wireless link for measured data transmission.

The demonstrator consists of two KETCubes. The first KETCube serves as a (USB-powered) smart sensor equipped with a textile capacitor and Texas Instruments FDC2214 capacitance to digital converter – see Figure 8. The principle of sensing is as follows: the textile capacitor is located at a place, where an object presence should be monitored. If an object with a certain weight is placed at the sensor plate, the capacity of the sensor changes accordingly. The capacity is measured by FDC2214 connected to the KETCube.

The second KETCube (Figure 9) serves as a PC-connected data concentrator. The concentrator continuously re-transmits data received from the sensor to a PC, where the measurement is visualized by a simple python script (a part of the KETCube project – support tools).

When powered up, the wireless link between both KETCubes is established using standard KETCube modules only (starNetwork module). The sensor site continuously transmits measured values and thanks to the wireless link between the two KETCubes, the data are on-line displayed on the connected PC.

The only parts specifically created for this project were the FDC2214 firmware module, the physical interconnection

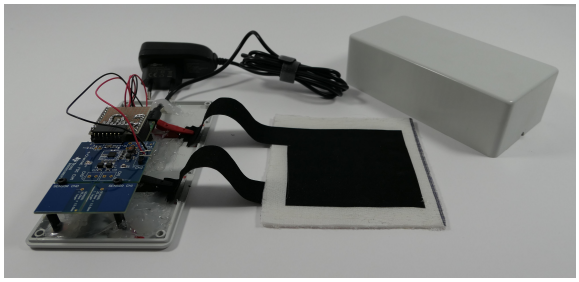

Fig. 8. Textile capacitive sensor connected to the KETCube in a custom box

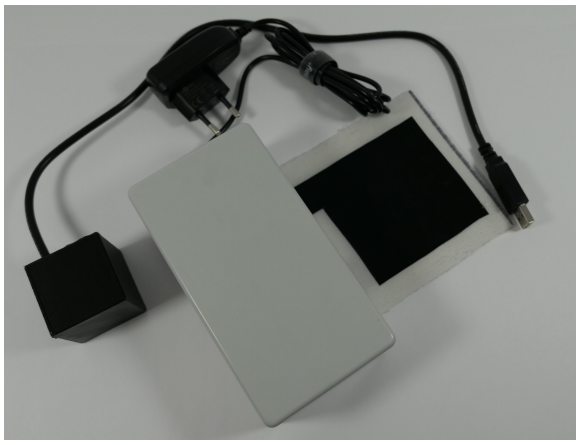

Fig. 9. Textile capacitive sensor with the KETCube in a custom box (white) and the KETCube as a data concentrator in the standard KETCube box (black) equipped with an in-cable FTDI Serial2USB converter

of FDC2214 development board and KETCube and the custom box.

The *Object Presence Sensing Demonstrator* is currently a work-in-progress project requiring additional work in the textile sensor testing and validation. If the project will move to the next phase, the textile sensor will operate autonomously and an LPWAN network will probably be used. Here, the advantage of the KETCube will appear: no additional work will be required, as only different KETCube module will be used for wireless communication and none of both – hardware and software – will require modifications.

C. Educational Use-Cases

The KETCube platform is already used in the educational process at our institution. A project solved by one of our students involves the implementation of the firmware module for the *Thunder click*[11] mikroBUS™sensor board. The Thunder click is intended to detect the presence and proximity of a potentially hazardous lightning activity. The hardware configuration is in Figure 10.

As a part of the work, an application note describing the deployment and use cases related to an IoT sensor equipped with the Thunder click will be created.

Direct usage of the KETCube platform in the educational process (in a class) is planed for the next semester.

[11] https://www.mikroe.com/thunder-click

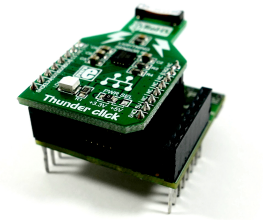

Fig. 10. *Thunder click* board in a mikroBUS™-compatible KETCube socket

IV. Conclusions

We identified challenges coming from a heterogeneous and fast growing area of IoT related to the sensor node development. Based on our experience in R&D and educational process, we proposed a novel KETCube platform intended to support both R&D and educational activities.

The current release of the KETCube platform includes the *main board*, *battery board*, *datasheet*, three *application notes* and *firmware* (v0.1).

The current release of KETCube platform Core firmware (v0.1) allows the module management and configuration. Enabling/disabling modules can be performed in the compile time – module linked or not – or in the run-time – module On/Off – by using a serial terminal.

The current release of the KETCube platform Modules firmware (v0.1) includes the LoRaWAN module (full Class A support), a proprietary starNetwork module (allows to form an ad-hoc KETCube network), the ADC module and the HDC1080 module. Thanks to the ADC module, the KETCube can be used in connection with any voltage output sensor.

The documentation currently associated with the KETCube includes the Datasheet and a set of basic application notes describing: (i) the KETCube module creation process, (ii) the voltage measurement up to 100V DC (theory of operation, front-end circuit and KETCube settings) and (iii) LoRa module (configuration and customization).

V. Future Work

Most of the future work currently concentrates on the KETCube firmware, although other tasks – e.g. new firmware/hardware modules – will be performed in parallel.

As the KETCube platform development itself started originally from a particular application-oriented project (based on STM32 and Semtech code), the code of (some) firmware modules still contains residues of a code, which requires refactoring.

A great challenge is the adoption of a test and validation methodology and their incorporation into the open KETCube project in an appropriate way.

The set of modules will be soon extended by the Sigfox module, which is currently under development.

The KETCube is/will be used in certain projects (co-)solved in our department including the development of autonomous sensor for the Fire Rescue Service of the Czech Republic distributed by a quadrocopter and development of a custom environmental sensor for *smart farming*.

Currently one of our industrial partners started a KETCube-based smart-metering IoT node development project, which includes proprietary development, but it will also result in contributions to the public KETCube project.

Acknowledgment

This research has been partially supported by the grant QK1810010 of the Ministry of Agriculture of the Czech Republic, "Automatic system for collecting and processing of temperature and humidity parameters of microclimate and soil for conditions of precision farming in the Czech Republic on the principle of the Internet of Things (IoT)" (2018 – 2022).

References

[1] T. Snyder and G. Byrd, "The Internet of Everything," *Computer*, vol. 50, no. 6, pp. 8–9, 2017. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MC.2017.179

[2] S. Musa, "Smart cities – a road map for development," *IEEE Potentials*, vol. 37, no. 2, pp. 19–23, March 2018.

[3] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 855–873, Secondquarter 2017.

[4] J.-P. Bardyn, T. Melly, O. Seller, and N. Sornin, "IoT: The era of LPWAN is starting now," in *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*. IEEE, 2016, pp. 25–30.

[5] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[6] Y. Xia, "Cloud control systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 134–142, 2015.

[7] J. He, D. C.-T. Lo, Y. Xie, and J. Lartigue, "Integrating internet of things (iot) into stem undergraduate education: Case study of a modern technology infused courseware for embedded system course," in *Frontiers in Education Conference (FIE), 2016 IEEE*. IEEE, 2016, pp. 1–9.

[8] "The University of Illinois/NCSA Open Source License (NCSA)," https://opensource.org/licenses/NCSA, accessed: 2018-04-11.

[9] "mikroBUS™," https://www.mikroe.com/mikrobus, accessed: 2018-04-11.

[10] "STM32L0 Series," http://www.st.com/en/microcontrollers/stm32l0-series.html, accessed: 2018-04-11.

[11] "Semtech SX1276," https://www.semtech.com/products/wireless-rf/lora-transceivers/SX1276, accessed: 2018-04-11.

[12] "Type ABZ," https://wireless.murata.com/eng/products/rf-modules-1/lpwa/type-abz.html, accessed: 2018-04-11.

[13] "HDC 1080," http://www.ti.com/product/HDC1080, accessed: 2018-04-11.

[14] *KETCube datasheet*, University of West Bohemia in Pilsen, 2018, rev. 05/2018, https://github.com/SmartCAMPUSZCU/KETCube-docs/blob/master/KETCubeDatasheet.pdf.

[15] D. Boswell and T. Foucher, *The art of readable code*. " O'Reilly Media, Inc.", 2011.

[16] "KETCube AppNotes," https://github.com/SmartCAMPUSZCU/KETCube-docs/tree/master/appNotes, accessed: 2018-05-10.

[17] "GNU/Indent," https://www.gnu.org/software/indent/, accessed: 2018-04-11.

[18] "Doxygen," https://www.doxygen.org/, accessed: 2018-04-11.

[19] "Open Source Hardware Documentation Jam," https://www.oshwa.org/2013/03/17/open-source-hardware-documentation-jam/, accessed: 2018-04-11.