Event-driven kinematic measurements using BBC micro:bits programmed in C++

SOMOGYI Anikó, KELEMEN András, MELLÁR János, MINGESZ Róbert

Abstract In Hungary, according to the Physics curriculum in National Core Curriculum 2020, quantitative description and examination of motions are required. Teachers and students are suggested to use the available computer tools and programs during the measurements and evaluation. We present and recommend microcontroller-driven measurements displaying the physical quantities of the mentioned movements as a function of time. The measurement setup consists of a linear air track, BBC micro:bit, some optical sensors and 3D-printed elements. The event-driven measurement software running on the micro:bit was developed in C++ (CODAL). The microcontroller sends measurement data to the PC through wireless communication (radio or BLE), and the Excel Data Streamer and a self-developed add-in for Excel serves to receive and display real-time data. Linear and non-linear regression analysis of measurement data was performed by built-in Excel functions, Trendlines and the Solver add-in. These measurements can also add interesting extra material to the curriculum in IT classes.

Keywords: BBC micro:bit, event-driven measurement, simple motions, optical sensors, serial communication, Excel Data Streamer

1. Introduction

In the 16th century, the examination of movements with scientific rigour became a natural science based on quantitative measurements. Galileo formulated a precise relationship between time and the displacement during the free fall of an object, also he slowed down the motion using a slope [1]. He also discovered that the period of a simple pendulum is independent of mass [2]. These simple motions form the basis of the topic of high school mechanics, including kinematics, even today [3]. The effect of his research method can be observed even after hundreds of years: The so-called scientific method remains an important part of the National Core Curriculum 2020 of science education in Hungary. One of the most important steps in the process is experimental data collection and analysis, which in the 21st century can be implemented not only with traditional measurement techniques but also with modern tools of technical informatics. The curriculum also recommends the use of available computer tools and programs in measuring and evaluating them [4]. This innovation is to be welcomed because, on the one hand, it provides an opportunity for a kind of interdisciplinary education, i.e. one that treats computer science and science as a whole, and, on the other hand, for student activity.

At the same time, the topic of robotics plays a significant role in the education of digital culture, which, of course, includes sensory measurements [4]. In this way, a kind of fusion of physics and computer science can be realized in education. In digital culture textbooks [5–8], the programming of the BBC micro:bit device, which is very popular and recognized internationally and designed specifically for educational purposes [9–12], plays a prominent role in this topic, one of the probable reasons for which is that a possible temporary shortage of the physical devices can be partially skipped using MakeCode, or the micro:bit Python Editor simulator [13,14]. Many studies can be accessed by interested teachers and students on the website of the program called 'Stumbling of micro:bits' organized by ELTE Faculty of Informatics[15], among which their after-school program curriculum [16] stands out. With the help of these, one can learn the basic use of the built-in sensors of micro:bit V2 (brightness, sound, acceleration, temperature, magnetism sensor), so they can also help those involved in physics education.

The HAS Research Group of Technical Informatics Methodology, in which the authors of this article also participate, among other things aimed to develop methods and tools to support students in learning the basics of engineering thinking through computer-aided measurements and programming easily accessible microcontroller-based devices (Arduino, BBC micro:bit) [17]. In addition to this, several experiments have been developed and published, which can be included in the classroom following the curriculum requirements described above [18–20]. On the website of the research group, both physics and computer science teachers can find aids in Hungarian for the inclusion of technical information technology content in their subjects [21].

In addition to the recording of the measured data, the precise evaluation of the data is of particular importance. The spreadsheet program Microsoft Excel has been among the software recommended by subject pedagogy researchers who wish to modernize the methodology of physics education with the inclusion of computers for a long time. [3,22,23]. Several publications have been published showing how to connect the BBC micro:bit or Arduino with Excel in computer-aided measurement experiments. [24–29].

In this study, we present for the first time an experimental setup based on the BBC micro:bit, containing 3D printed and homemade elements. Next, we describe the measurement algorithm and its software implementation written in C++, running on the micro:bit. and an Excel add-in (Data Streamer) for receiving the measured data on a PC. The evaluation is carried out using Excel's built-in functions and extensions. This hardware-software combination can be suitable for the application of microcontroller-controlled measurements in education, including the computer-aided, quantitative examination of kinematic phenomena prescribed by the framework curriculum.

2. An experimental setup based on the BBC micro:bit

In addition to the built-in sensors of the BBC micro:bit, additional sensors can be connected to the micro:bit. At first, sensors can be connected to wider pins (0, 1, 2) on the edge connector of the micro:bit with alligator clips, but using a panel, a so-called breakout board with a simple row of pins (or possibly with pin holes) is recommended [30], with the help of which circuit elements can be easily and stably connected to all I/O connectors (even the narrow ones), which is not possible with wide alligator clips.

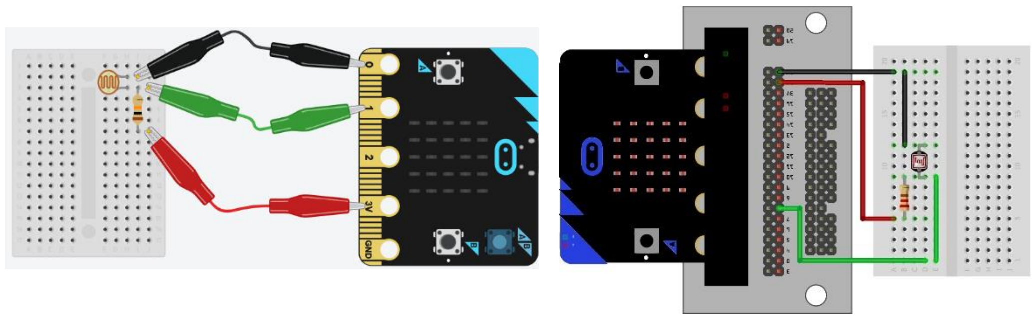


Figure 1. Connecting a photoresistor and resistor in series on a breadboard and wiring them to the edge connector of the BBC micro:bit a) with crocodile clips[1], or b) using a breakout board[2]

[1] The figure was created in the Tinkercad simulator.

[2] The figure was created in the Fritzing application.

2.1. BBC micro:bit supplemented with a reflective photosensor (line tracking sensor)

The continuous detection of the position of a body moving in a straight line can be realized with an experimental setup that works analogously to the quadrature encoder transmitters used to measure angular rotation [31,32]. In the first step, we can stick a self-adhesive encoder tape with alternating black and white (IR absorbing and reflecting) matte surfaces to the path of the movement (in our case, to the bottom of the side wall of a linear air track[3], under the rows of air blower holes), we have attached an A4 printable design for this [34]. A micro:bit and a reflective IR transceiver (e.g. TCRT5000-M line tracking sensor module) connected to one of its digital inputs can be placed on the cart so that the sensor is positioned above the tape during movement. The output signal of the sensor is thus a binary square signal: there are edges in the signal at the borders of the black and white fields (at every 0.785 cm). When these events occur, the position as a function of time can be measured together by recording the time stamp and incrementing the position counter.

If the direction of the movement also changes (so the position counter should be decremented) during the movement, adding another sensor to the setup is advised. When the two sensors are placed at a distance appropriate to produce signals with a phase shift of about a quarter of a period, the so-called quadrature signal appears on their output (Figure 2). Deciding which signal follows the other one, makes it possible to detect movement direction and position. A similar experiment rail equipped with an encoder tape and a small cart equipped with sensors and a controller [35] is available commercially [36], but this assembly can be implemented at a fraction of the cost.

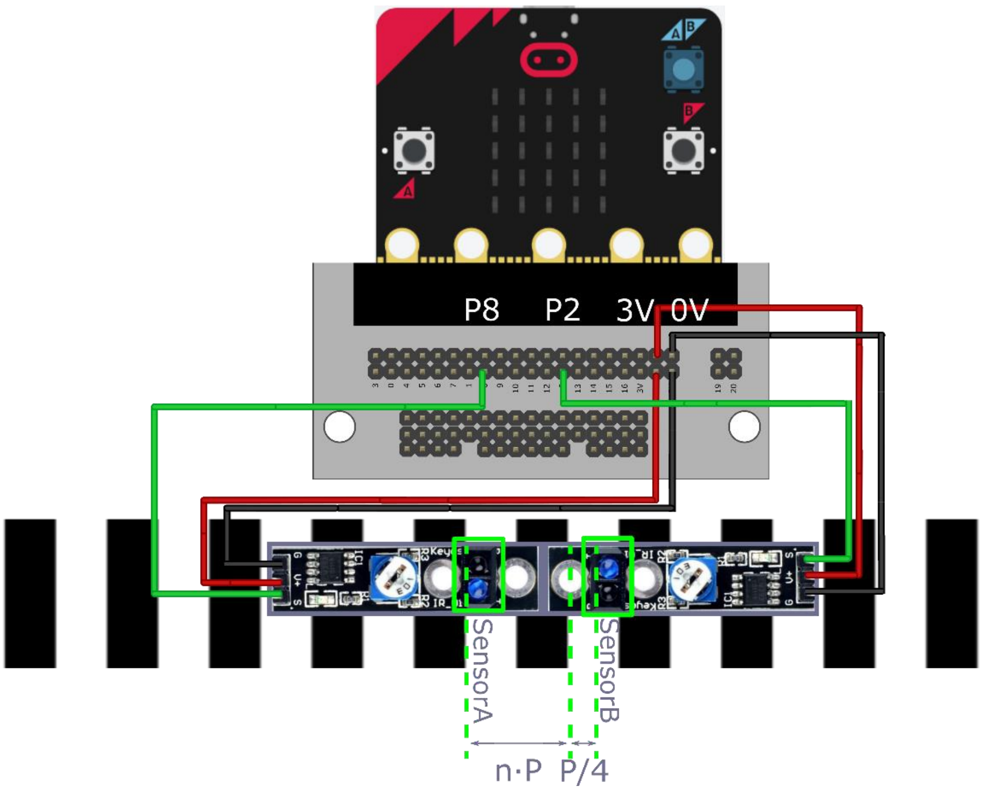


Figure 2: A circuit consisting of the TCRT5000-M modules and the BBC micro:bit. Sensor modules placed above the tape with distance of $n \cdot P + \frac{P}{4}$, where $P = 1.58\ cm$, $n \in \mathbb{N}$ (and in this figure n=1), produce a square wave signal with a phase difference of a quarter of a period

[3] A low-cost DIY version can be made instead of air tracks available commercially [33]. We tested our experiments at the SzeReTeD student laboratory of the University of Szeged.

2.2. Implementation of wireless data transfer between the micro:bit and the computer

Since we designed a wireless measurement experiment on the air track, there is naturally a need to implement a simple microcontroller system moving with the cart to ensure sending the logged data to the computer. The micro:bit card is controlled by the Nordic Semiconductors brand NRF52833 (BLE) [37], which also enables wireless communication for the device as a built-in radio frequency module.

One possibility is based on radio communication of two micro:bits, the circuit of the moving micro:bit can be mobilized with real-time data transmission [38–40]. The micro:bit placed on the cart serves as a wireless measurement system and also as a radio transmitter of measurement data, while the other device as a radio receiver is connected to the PC via USB.

Another option is to establish a Bluetooth Low Energy connection by pairing the micro:bit with a suitable computer. In this way, the wireless transmitter moving on the rail can send the measured data directly to the computer.

These two ways we can implement wireless data transmission between the measurement device and the computer [41,42].

For experiments on an air track, it is needed to find a power supply that is not too heavy for the air track experiments and can provide the system with the necessary current intensity at 3V operating voltage so that the operation of the sensors and communication would work reliably. Therefore, it is not advisable to power the transmitter with 2 heavy, 1.5V AAA batteries connected in series, though those would provide enough intensity of current. We also dismissed the possibility of using a CR2032 3V coin cell despite its lightweight, since its current output capabilities are too small to be used with the system the low intensity of current. The small, 190 mAh rechargeable Li-Po battery (3.7-4.2 V) would match the two criteria above, however, it cannot be connected directly to the power socket based on the circuit specification. However, the micro USB connector is equipped with a suitable voltage regulator, through which the battery power can be connected, e.g. with the help of a JST-USB adapter obtained by soldering a line of pins to a broken micro USB cable, so we can provide a stable and light-weight power source for the micro:bit that moves with the cart. The block diagram of the setup providing communication and power supply is shown in Figure 3.

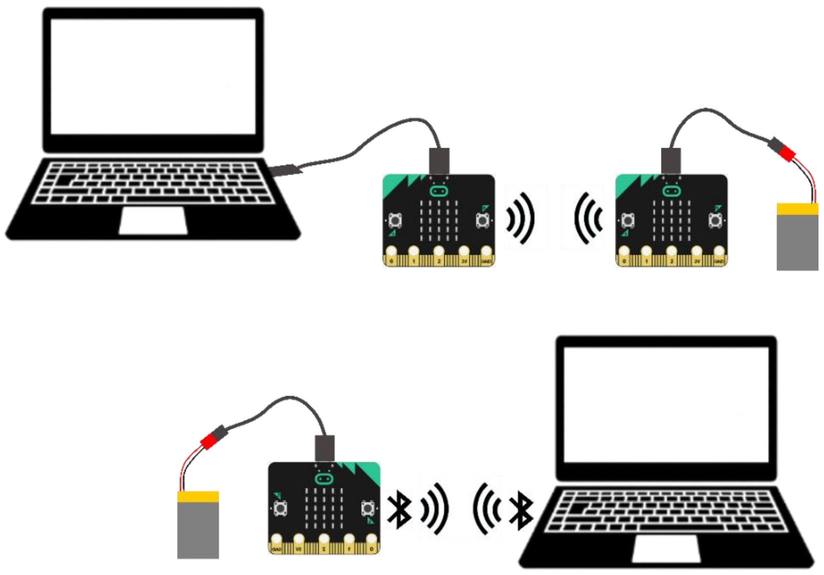


Figure 3: Block diagram of wireless data transmission (radio communication above, BLE communication below)

2.3. A 3D-printed cart moving on a linear air track for the micro:bit and its accessories

3D-printed devices are becoming more and more popular in science education as well, including physics laboratories, which can be enriched with self-designed devices [43]. We can design (e.g. in Tinkercad or Fusion 360) a unique cart that is suitable for the stable carrying of the above circuit, i.e. it provides a suitable place for the micro:bit, its battery and the sensor modules. In the case of the latter, special attention must be paid to the fact that, on the one hand, they are located above the encoder tape, and on the other hand, the sensors must be positioned to produce a quadrature signal.

It is worth considering a few more practical aspects when designing. It is recommended to reduce the air resistance as much as possible, by rounding the edges. The air coming out of the rail must be able to lift the cart away from the wall of the rail, thus one aim is to limit its mass, and on the other hand, the appropriate mass distribution for balance and stability must be ensured. For stability, it is worth designing the side walls hanging from the rail to be long enough. By placing the sensors on the same sidewall, our cart may become unbalanced, which can prevent the airflow on that side, so it is advisable to place internal long cavities in the walls. By throwing small metal balls into these, the object can be balanced. At the inner edge under the spine of the cart, it is also worth making a small longitudinal hole (approx. 1 mm in diameter) to ensure airflow, which can significantly reduce friction. We have attached the 3D design of the cart, designed in Fusion 360 together with our students [34].[4]

The complete experiment setup, with which the position-time dependence of rectilinear movements can be examined, can be seen in Figure 4.

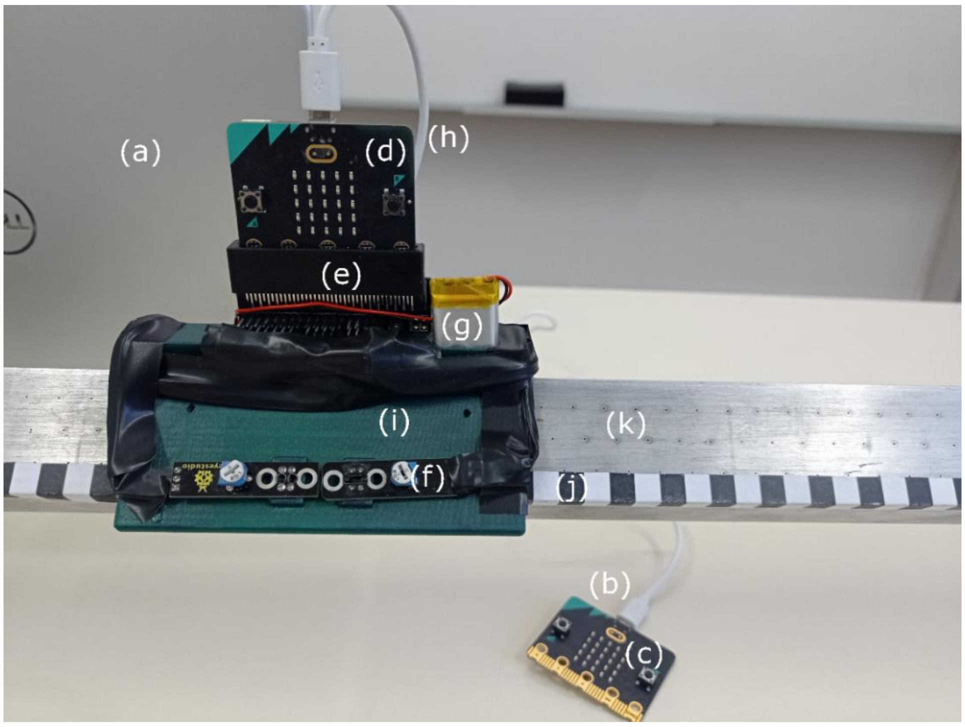


Figure 4: An experimental setup for position-time measurement of movements on an air track:
a) PC, b) micro USB cable, c) wired micro:bit (radio receiver), d) wireless micro:bit (radio/BLE transmitter)
e) breakout board, f) reflective IR transceiver (A TCRT5000-M), g) Li-Po battery
h) micro USB adapter, i) 3D-printed cart j) encoder tape k) linear air track

[4] Since the encoder tape must be glued to the air cushion rail in such a way that the air flow is not disturbed, the length of the side walls and the location of the sensors might be modified when using our model.

3. Software development for the implementation of kinematic experiments based on the BBC micro:bit

We see far fewer examples of measurement experiments with the micro:bit in the literature than with other popular platforms, like Arduino. Therefore, we considered it important to describe the structure of the software tools used for the experiments in detail.

3.1. Programming a micro:bit as a measurement device

In Section 2, we described the structure of the experimental setup, from which it was also revealed that it is necessary to decode square signal(s) to be able to determine time and position. We developed a basic version of the measurement program suitable only for investigating one-directional movement and then a more complicated version for motion with alternating directions. Both versions were created for both radio communication and BLE communication as well. We provide the complete program codes in the supplementary of this paper (both the C++ codes and the .hex machine codes as well) [34].

3.1.1 Programming the micro:bit in different environments

BBC micro:bits can be programmed with block programming on the MakeCode web application [44], or in JavaScript [45] or Python (MicroPython) [14]. It is perhaps less well-known that a wide range of function libraries are also available in the Arduino (C/C++) language [46]. Both MakeCode and MicroPython are based on the DAL/CODAL abstraction layer written in C++, which is based on the NRF5 SDK [47,48]. In general, it can be said that the lower-level language the device can be programmed in, the greater time accuracy can be achieved [49]. For the current experiments, the measurement software was developed in the C++ language using CODAL which supports micro:bit V2 [50].[5]

3.1.2 Handling events and scheduling tasks in C++ language using CODAL

In CODAL's C++ code, the `main()` function we place setting-type commands to be executed once e.g. we initialize the instance of the micro:bit class typically called ‘`uBit`’, we set our digital pin to input by calling once the `getDigitalValue()` function and also set the pull mode of this input to pull-up.

In CODAL, it is possible to implement a kind of multithreading, on the one hand, through its non-preemptive fiber scheduler, which enables the coordinated operation of several concurrent tasks, and on the other hand, through the event management system called message bus, which allows writing reactive code The scheduler is initialized for `uBit` by calling `scheduler_init(uBit.messageBus)` [47].

MessageBus serves on one hand to record an event that occurs in a specific hardware block (e.g. button press or loud sound detection, or the arrival of a radio message), and this event is monitored at runtime. The MessageBus also delivers the occurring events to the program through event handlers. By calling the `listen()`, we attach a callback to a function when a specified event occurs.

[5] Installation instructions and sample codes can be found on Lancester University's GitHub page [51]. The compiled .hex files must be copied to the micro:bit driver.

We applied one of the built-in pushbuttons of the micro:bit as a start button. When the MessageBus sends a notification of the click of the button, the attached event handler is called to change the value of the '`toggle`' Boolean variable.

Similarly, in our experiment unit jump (falling or rising edge) events occur in the digital signal (`MICROBIT_PIN_EVT_FALL` or `MICROBIT_PIN_EVT_RISE`, respectively) on one of the digital inputs (e.g. `MICROBIT_ID_IO_P2`). We note that listening to these events must be enabled on the specified pin by `uBit.io.P2.eventOn(MICROBIT_PIN_EVENT_ON_EDGE)`. The MessageBus informs the program, and the scheduler calls the customizable event handler function (`onEdgeA`). Without setting the scheduler, all events have the same priority, however, after calling the scheduler, we can highlight these time-critical events with the keyword `MESSAGE_BUS_LISTENER_IMMEDIATE`.

The code fragment that implements the monitoring of the event:

```
  uBit.messageBus.listen(MICROBIT_ID_IO_P2, MICROBIT_PIN_EVT_FALL,
  onEdgeA, MESSAGE_BUS_LISTENER_IMMEDIATE);
```

By calling `create_fiber(sendAvailableData)` once in the `main()` function another fiber is initiated and its previously defined parameter function is called from this fiber by the scheduler. To help concurring fibers, it is advised to use `fiber_sleep(delay_millisec)` function in this non-time-critical fiber to let the scheduler run the time-critical event handlers. The main function also represents a fiber which in our case is released after initializing every component of the program.

3.1.3 Producer and consumer - writing and reading a circular buffer

One possibility to process data in the created fiber would be to check whether the currently available timestamp differs from the one stored when the previous event occurred because then we can know that a new event has occurred [28]. However, when the cycle time of the process is longer than the timespan between consecutive events, timestamps for some events may not be processed. To avoid that, we defined an array of timestamps serving a circular buffer.

The event handler assigned to the rising edge events represents the producer task, storing measurement data at the index called '`writePointer`'. The created fiber for data process and transmission is used as the consumer, reading data from the buffer at the index called '`readPointer`'. To imply a circular boundary condition, we set these indices back to zero when reaching the end of the array. The program also monitors the number of available samples: when an event occurs, its callback function writes a piece of data increasing the variable for the number of available samples, but when the `sendAvailableData()` is called by the fiber scheduler, the same variable is decreased.

The code fragment that implements the event handler (producer):

```
void onEdgeA(MicroBitEvent evt)
{
    if(toggle)
    {
        dataBuffer[writePointer]=evt.timestamp;
        writePointer++;
        if (writePointer == BUFFER_LENGTH)
        {
            writePointer=0;
        }
        if (availableSamples < BUFFER_LENGTH)
        {
            availableSamples++;
        }
        else
        {
            processingTimeout = true;
        }
    }
}
```

When the elapsed time between two events is less than the process time, the number of available samples increases dynamically. When the buffer is full, i.e. number of available samples equals the buffer size, the timeout is handled by stopping the producer (setting the toggle flag to false), emptying the buffer by the producer, and afterwards freezing the program by an infinite loop [20].

The code fragment that implements the consumer and handles timeout:

```
void sendAvailableData()
{
    while(1)
    {
        if(availableSamples)
        {
            if (processingTimeout)
            {
                uBit.display.scrollAsync("TIMEOUT");
                toggle=false;
                if(availableSamples==0)
                {
                    while(1);
                }
            }
            timeStampData = dataBuffer[readPointer];
            …
            readPointer++;
            if (readPointer == BUFFER_LENGTH)
            {
                readPointer=0;
            }
            availableSamples--;
        }
        fiber_sleep(1);
    }
}
```

3.1.3. Time measurement in the event handler function

Although it is possible to sample the digital signal at equal intervals, only those events provide valuable information about the position of the body as a function of time, when a jump (rising or falling edge) occurs in the digital signal. If at this moment our event handler captures the timestamp (`evt.timestamp`) with microsecond accuracy, we assign a time to a very well-defined moment of the movement: the cart is switching between neighbouring black and white fields of the tape.

Since the event handler function should be executed as quickly as possible (since the scheduler handles it as time-critical), it is advised to keep it as short as possible. Further operations on the measured time data and its transmission to the PC will take place in an infinite loop placed in the fiber of the consumer. The timestamp in CODAL is an unsigned 32-bit integer, so reading data from the buffer is an atomic operation for the 32-bit processor of NRF52833.

3.1.4. Measuring the position of one-way movement by counting events

When processing the signal of the reflective IR transceiver sensor module, in addition to recording the timestamp, we also want to assign a position value to the event. Since the width of the white and black stripes on the encoder tape is the same, it is advisable to record the timestamp for both the rising and falling edge events, and these can be handled alternately with the same event handler routine. In the case of one-way movement, each new edge means an increase of the location coordinate by 0.785 cm, so in the endless loop of the consumer (in addition to processing and sending the timestamp), the position counter variable is also incremented.

3.1.5. Measuring the position of alternating direction movement by decoding quadrature signal

The direction and then the position of the alternating movement can be decoded from the 4 states of the quadrature signal measured by the two sensors, shifted by approximately ¼ period (Table 1).

The logic behind our method of decoding the quadrature signal is as follows. The event handler of one signal (A or B) can derive the information of the edge type, that is falling or rising from the `evt.value` variable, from which we can deduce the binary state of the particular sensor (A or B) right after the edge event. By changing the value of a binary flag respectively, each time, the signal (A or B) can be reproduced.

The code fragment that implements detecting Signal B:

```
void onEdgeB(MicroBitEvent evt)
{
    if(evt.value==MICROBIT_PIN_EVT_FALL)
    {
        stateOfSensorB=0;
    }
    else
    {
        stateOfSensorB=1;
    }
}
```

Ideally, we could ensure with sufficient geometric accuracy that there is exactly a ¼ period shift between the two signals. Then, a timestamp could be recorded in events fired by both types of (rising and falling) edges of the signals of both sensors (A and B), between which the position would change by 0.3925 cm in the positive or negative direction. The inaccuracy of 3D printing and the size of the sensor modules do not allow this level of accuracy. Therefore, as a compromise, we still record a timestamp only at the edges of the signal A, the distance travelled between consecutive events is 0.785 cm. Instead of the one-dimensional array, we applied a two-dimensional one for the circular buffer to store the belonging timestamp, the state of signal A and the state of signal B.

Based on Table 1, it is shown that if the binary state of signals after the edge events is the same, signal A is followed by B, otherwise, the direction of movement is reversed. In the code, we also assign a binary value to the movement direction and record this flag.

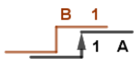
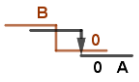
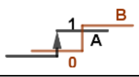
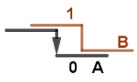
Changes of signal A (Events)	Type of edge	State of signal A after the event	State of signal B after the event	Direction of movement and change of position
	Rising	1	1	A is followed by B↦1
	Falling	0	0	A is followed by B ↦1
	Rising	1	0	B is followed by A ↦0
	Falling	0	1	B is followed by A ↦0

Table 1: Events in the quadrature signal and the logical structure of the decoding

Table 2 illustrates the conditions for incrementing and decrementing the position. The movement direction established in the current event must be compared with the movement direction recorded in the previous event, and if it has changed, the position counter must be increased or decreased.

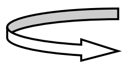
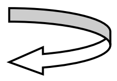
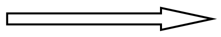
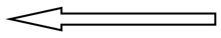
Movement phase	Movement direction at the previous event	Movement direction at the current event	Change of position
	B is followed by A ↦0	A is followed by B ↦1	No change
	A is followed by B ↦1	B is followed by A ↦0	No change
	A is followed by B ↦1	A is followed by B ↦1	Incrementation
	B is followed by A ↦0	B is followed by A ↦0	Decrementation

Table 2: Logical structure of position counting

3.1.6. Radio communication between two micro:bits

In the code of the micro:bit that plays the role of the transmitter, in the first part of the main() function, where we place the commands that are executed once, we enable the radio connection (`uBit.radio.enable()`), and then in the infinite loop, after the type conversion of the timestamp and after changing the position counter properly, the time and position data pair is sent to the receiver as a `ManagedString` type message, separated by commas and provided with a newline character (`uBit.radio.datagram.send(message)`). On the other micro:bit - functioning as a receiver - we also enable the radio connection, and this time the `listen()` function monitors the event of the arrival of the radio message [39,51,52].

3.1.7. Serial communication between a micro:bit and a PC

When data arrives, the data is loaded into a buffer, which is converted into a `ManagedString` type message and then sent to the PC via the serial port (`uBit.serial.send(message)`). Note that the default baud rate of communication, in this case, is 115200 bit/s [53].

3.1.8. Bluetooth communication between a micro:bit and a PC

CODAL's Bluetooth capabilities are currently under development by staff at Lancaster University. As of now, quite a few services are available, e.g. the Event Service, the IOPin Service, the Button Service, Magnetometer Service, etc. For wireless data transmission, we instantiated the UARTService ('`uart`' pointer) [54]. According to its library, it is possible to send ManagedString type messages as well as 8-bit unsigned integer pointers (byte arrays). Another parameter of the `uart->send()` function is the mode, which can be SYNC_SLEEP or ASYNC. If we use the previous one, the function will perform a cooperative blocking wait until all given characters have been received by the connected device, thereby slowing down the fiber. For a multi-task fiber, one should choose the ASYNC option, which will copy as many characters as it can into the buffer when called and then return control so further commands can be run immediately, while ASYNC task runs in the background. If the data transmission is not possible for some reason (the buffer is full or the reading is delayed), the send function returns a negative error code, otherwise with the number of successfully sent bytes [55]. To ensure that each character is sent sooner or later, we iterated the send command using a for loop:

The code fragment that implements sending a string message in asynchronous mode:

```
int uartSendAsync(ManagedString string)
{
    int length = string.length();
    int sent = 0;

    while ( sent < length)
    {
        int bytes = uart->send( string.substring( sent, length - sent), ASYNC);
        if ( bytes >= 0)
        {
            sent += bytes;
        }
        fiber_sleep(1);
    }
    return sent;
}
```

In both versions of our measurement program (adequate for one-directional or two-directional motions) a timestamp and a position value with a separator character and a newline character are sent to the PC. Since timestamps in microseconds are 8-9 or even more characters long, these messages turned out to be too long and the Bluetooth transfer rate slowed down. Messages can be shortened by sending both pieces of information as whole numbers (4 bytes) without extra characters. Conversion of an integer is implemented by bitwise operations as seen below [56]:

The code fragment to convert integer to byte array:

```
int uartSendAsyncByteArray(int32_t num)
{
    uint8_t numByteArray[4];

    numByteArray[0]=(num>>24)&0xFF;
    numByteArray[1]=(num>>16)&0xFF;
    numByteArray[2]=(num>>8)&0xFF;
    numByteArray[3]=(num>>0)&0xFF;
      …
}
```

3.2. Receiving data on PC

During the kinematic measurements, the micro:bit sends time-position data to the computer as it was described previously. In the radio communication version the receiver micro:bit sends data via USB to the computer through Serial COM Port. When data is sent through BLE, the micro:bit serves as a GATT Server and our self-developed add-in serves as a GATT Client to receive wireless data.

3.2.3. Serial data reception with the Excel Data Streamer add-in

There are several ways to receive serial data on a computer, the simplest of which is to use the serial monitor of a development environment (e.g. Arduino IDE, Visual Studio Code). Another solution is to use free add-ins for Excel, like PLX-DAQ and our choice, Microsoft Data Streamer for Excel, which is an optional and free COM add-in available for Excel users that becomes more popular in STEM projects because it makes it possible to evaluate the received serial data immediately, locally [24,29,57].

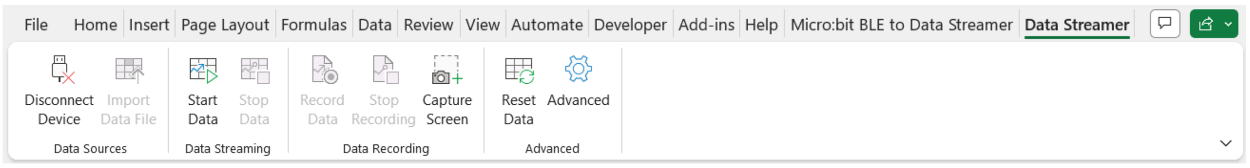


Figure 5: Microsoft Data Streamer for Excel add-in on the ribbon menu

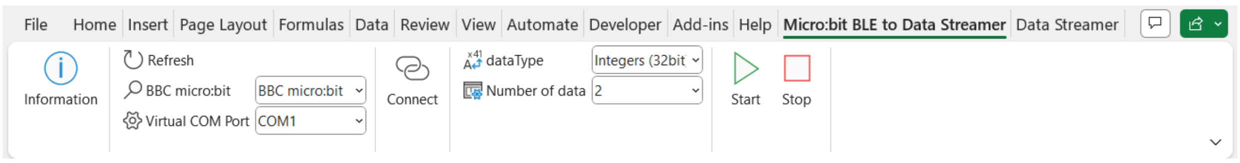
The settings of the measurement software and the data-receiving add-in must be coordinated. In addition to adjusting the previously mentioned Baud Rate (in our case 115200 bit/s), we can specify how many pieces of data separated by commas (time and position values, so Channels number is 2) are received per line. In the current version of the program, the number of lines to be stored can be increased up to one million (the default number is 15).

3.2.3. Data reception with the Micro:bit BLE to Data Streamer add-in

One can find very few available software solutions for wireless data transfer from the micro:bit to a computer. In most applications, it is understandable that developers rather concentrate on data receiver applications on Android or iOS platforms, since for student experiments or IoT projects these are more adequate. However, these experiments on the air track are demonstration experiments, and the aim is to show measurement results frontal to a class. We decided to develop an MS Excel add-in as a supplementary application for Excel Data Streamer.

Microsoft provides several options for developers to create an Excel add-in. One of the most common and oldest options is macro development in the Visual Basic language. Visual Studio Tools for Office (VSTO) provides the opportunity to develop offline extensions based on the Windows operating system in the Visual Studio environment, in the .NET Framework, in both C# and Visual Basic languages. Another possibility is the programming of online, platform-independent Office Add-in extensions in JavaScript/TypeScript.

We developed our data-receiving extension in C# using the VSTO toolbox. [58] The front-end page of the extension is the Ribbon built into Excel, which contains the buttons we specify, dynamic and static menus, drop-down lists (combo boxes), text fields, etc. There is a visual designer to design the Ribbon, but to access advanced functions (e.g. to write the code for dynamically expanding menus based on queries) we used the XML descriptive language. [59]On the backend page, in C# language, in the Ribbon class, we specify the event handler (callback) functions that will be called during the selection of various button presses and menu items (e.g. onAction, onPressed type event handler callback functions). The flexibility of the extension is made possible by the dynamic management of buttons and menus (e.g. getPressed, getLabel, getItemLabel, etc.)[60–62]. The .NET framework allows, for example, to create pop-up windows for more detailed setting options, or to create WinForms dialogs that convey warning or error messages. The user interface of our add-in can be seen in Figure 6.



BBC micro:bit: BBC micro:bit
Virtual COM Port: COM1
dataType: Integers (32bit
Number of data: 2

Figure 6: Micro:bit BLE to Data Streamer self-developed add-in on the ribbon menu

To communicate from Excel with the micro:bit, the Bluetooth Low Energy in Universal Windows Platform Apps technology was applied [63]. When the micro:bit is programmed to send data by Bluetooth UARTService, the micro:bit as a GATT Server writes its Transfer GATT Characteristics. The UUIDs of both the service and the characteristics can be found in the Bluetooth Profile of BBC micro:bit [64]. To pair and connect the device, find its UARTService and create a connection to read its TX characteristic, i.e. to create a Bluetooth GATT Client in the backend of our add-in, we followed the official tutorial of Microsoft [65]. When the GATT Characteristic's ValueChanged event is fired, the event handler reads from the transfer characteristic of the micro:bit to an argument buffer, from which a specified number of bytes, integers, strings, etc. can be read by an instance of the Windows.Storage.Streams.DataReader class. Depending on the user settings, received string messages, or four-byte integers converted to string are transferred to Data Streamer. We used a free virtual COM Port software to create a local bridge between the two add-ins [66]. We note that there is a possibility to read Excel cells right from a VSTO add-in, however, Data Streamer is optimized to write collected data to larger cell ranges at the same time with a user-defined rate. The other point of view is that it is advised to keep the ValueChanged event handler as short as possible to speed up Bluetooth communication, in our case received data is sent right after in the serial buffer, and no other data process tasks take place. We provided a downloadable installer for our add-in in the supplementary [34].

The running add-in has an Information window to inform users about connecting their micro:bit. By pressing the Refresh button, a query starts to list all available Bluetooth device paired with the PC, from which the BBC micro:bit can be selected. Similarly, all available COM Ports are listed, among which one-half of the previously created pair of Virtual COM Ports (local bridge) can be selected (while the other half must be selected for Data Streamer). Afterwards, connection can be asked, when the state of the Bluetooth device in Device Manager turns from Paired to Connected and the chosen

Virtual COM Port is opened. Users can choose which type of ValueChanged event handler is preferred according to the message format of the programmed micro:bit (ManagedString or integers). Also, the program needs a clue as to how much data is to be expected per line (in harmony with the Data Channels setting in Data Streamer). Afterwards, the press of the Start button results in reading TX characteristics by subscribing to notifications of its Indicate property and by starting the ValueChanged event handler.

3.4. Evaluation of the results of kinematic experiments in Excel

The measurement experiments were carried out on an air track and then evaluated in Excel. Our results show good agreement with the theoretical position-time correlations of the movements listed in the introduction.

3.4.1. Linear regression analysis of the measurement data of the experiments in Excel using the Trendline

Position–time data pairs were visualised on XY-point graphs in Excel. The option called Trendline offers the possibility to fit linear, logarithmic, exponential, polynomial, and power function curves on data or apply moving averages. In addition, equations of these curves and statistical R^2 values can be visualised on the graph. Among the examined movements on the air track, the space-time dependence of the uniform motion in a straight line and uniformly accelerated straight-line motion can be analysed using the Trendline.

3.4.2. Linear regression analysis of measurement data using Excel's built-in statistical functions

The features characterizing simple straight-line motions needed to be determined, applying the well-known equations for the position-time dependence of these motions. Though the coefficients can be derived from the equations of the Trendlines, one cannot change the accuracy of those values. Based on the least-squares method, Excel provides several functions of which output values can be used to determine those coefficients, thus features of motions like the initial position (x_0) and the velocity (v) of the uniform straight-line motion, or the initial position (x_0), the initial velocity (v_0)and the acceleration (a) of the uniformly accelerated motion (Table 3) [67,68].

Before beginning the evaluation, data ranges are advised to be named dynamically [69] e.g. Named Ranges **x** and **t** are for position and timestamp values, respectively. Afterwards, these ranges can be referenced in Excel formulas as can be seen in Table 3 [70].

Type of motion	Equation	Parameters evaluated in Excel	
uniform motion in a straight line	$\boldsymbol{x} = v\boldsymbol{t} + x_0$	v	SLOPE(**x**,**t**)
		x_0	INTERCEPT(**x**,**t**)
uniformly accelerated straight-line motion	$\boldsymbol{x} = \frac{1}{2}a\boldsymbol{t}^2 + v_0\boldsymbol{t} + x_0$	a	2*INDEX(LINEST(**x**,**t**^{1,2}),1,1)
		v_0	INDEX(LINEST(**x**,**t**^{1,2}),1,2)
		x_0	INDEX(LINEST(**x**,**t**^{1,2}),1,3)

Table 3: Parameters, equations and formulas to analyse position-time data for straight-line motions. Bold letters refer to Named Ranges

In Figure 7, the linear position-time dependence of uniform motion in a straight line can be seen. To gain parameters of the fitted linear function, simply SLOPE() and INTERCEPT() functions can be used to evaluate the velocity and initial position, respectively.

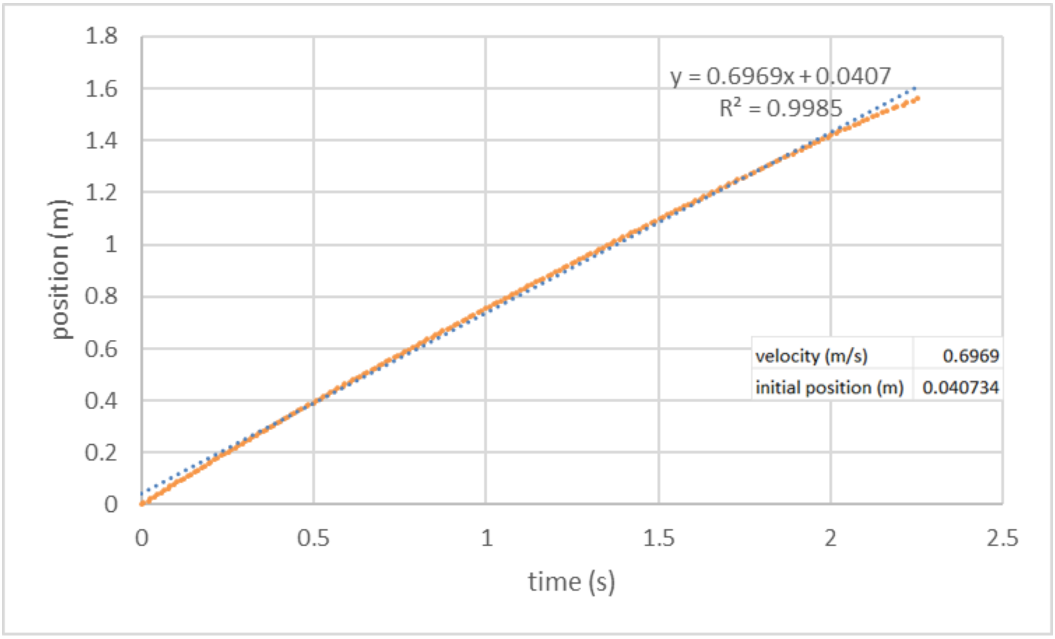
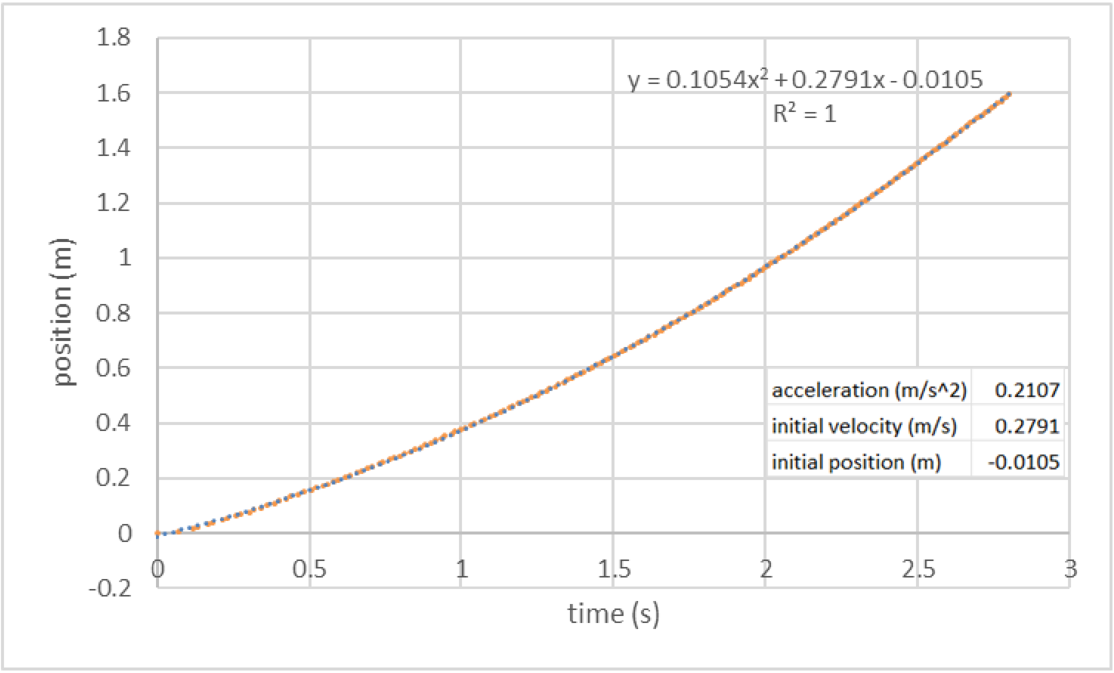


Figure 7: The position of the cart on an air track as a function of time in case of uniformly accelerated straight-line motion. Measurement data in orange and Trendline in blue are plotted.

While barely noticeable in Figure 7, we note that the time interval spent above a black stripe is not equal to the one above a white stripe. The reason for this is the fact that the analogue sensor signal edges are not quite sharp. A comparator on the module converts the signal to binary using signal cross-level detection, so it possibly happens that the edges in the signal emerge not exactly above the edges of the encoder tape. For more adequate position sensing one can choose to monitor events only on either falling or rising edges and compute position with a distance of 1.57 cm. Another solution can be to apply a moving average with a window size of 2.

We raised one end of the hover rail compared to the other end, so we got a slope with a small angle of inclination. Figure 8 illustrates that, as expected, a quadratic function describes the position as a function of time. The LINEST() Excel function can be used to extract the parameters of the quadratic function, which generates a table containing all the parameters and statistical characteristics of the regression. Using the INDEX() Excel function, we can select the coefficients of the members of the appropriate order of the fitted quadratic polynomial function from this data set [67].



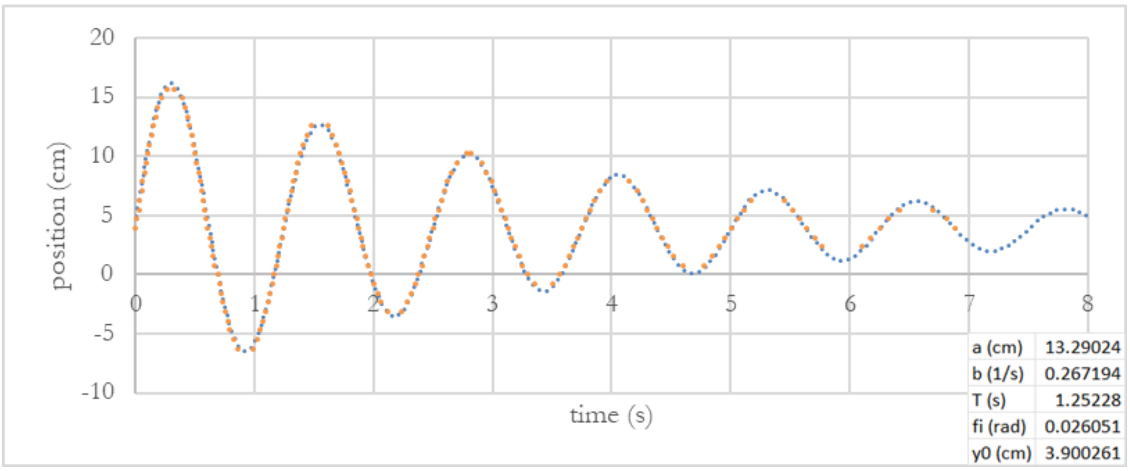
acceleration (m/s^2)	0.2107
initial velocity (m/s)	0.2791
initial position (m)	-0.0105

Figure 8: The position of the cart on an air track as a function of time in case of uniformly accelerated straight-line motion. Measurement data in orange and Trendline in blue are plotted.

3.6. Non-linear regression analysis by using Solver for least-squares optimization in Excel

In our modified experiment setup, the cart was attached between two relatively loose spiral springs. When we moved the object out of its equilibrium position, a damped vibration was created. When harmonic oscillation is damped by a force proportional to speed, the time dependence of the position can be estimated in the form of $y(t)=a\cdot e^{-bt}\cdot\sin\left(\frac{2\pi}{T}\cdot t+\varphi_0\right)+y_0$ [25]. It is not possible to fit a non-linear function like that using the previously used Trendline or the LINEST() built-in function. In recent years, step-by-step guides were published on how least-squares regressions can be applied to fit non-linear curves [71].

First, a, b, T, φ_0 and y_0 parameters were guessed as close to real as possible using the measurement data. Parameters a and b determine the exponential envelope, T means the period time of the oscillation, φ_0 is the initial phase, and y_0 is the offset of the damped sinusoidal function, i.e. the position of the equilibrium. We note that this estimation process might be very useful for the students helping them understand the meaning of each parameter. Second, with those parameters, the estimated y_{est} values, as well as the residuals ($y - y_{est}$) were counted. Then the sum of the squared residuals was determined and minimized by the Solver resulting in optimized values of the parameters. Using these parameters, a curve fit like the Trendline can be realized with the y(t) formula on a uniform, densely spaced time series as shown in Figure 8. We note that this more generalized method is suitable for the other types of motion, too.



a (cm)	13.29024
b (1/s)	0.267194
T (s)	1.25228
fi (rad)	0.026051
y0 (cm)	3.900261

Figure 9: The position of the cart on an air track as a function of time in case of damped oscillation. Measurement data in orange and Trendline in blue are plotted.

4. Summary

In the study, we presented a hardware and software assembly that, based on the BBC micro:bit teaching tool, can function as a complex measurement system for physics education. 3D printing played an important role in the experiment setup. During the presentation of the structure of the measurement software, we highlighted some of the possibilities inherent in the CODAL (C++) environment, regarding the potential inherent in event-driven programming. When evaluating the data, we used popular Excel extensions (Data Streamer, Solver) and functions (e.g. LINEST()). Our experimental results show a very good agreement with the theoretical description of the movements.

We believe that the tools and solutions presented above can also be useful for IT teachers: it is important to build relationships between subjects since education in these scientific fields can contribute to students choosing a career in the area of STEAM.

Acknowledgement

Supported by the ÚNKP-22-3 - SZTE-452 New National Excellence Program of the Ministry for Innovation and Technology.

Bibliography

1. S. Straulino, *Reconstruction of Galileo Galilei's experiment: the inclined plane*. Physics Education, (2008) 43. 316–321. http://www.fyysika.ee/vorgustik/wp-content/uploads/2011/11/Reconstruction-of-Galileo-Galilei.pdf (Last retrieved 07/27/2023). https://doi.org/10.1088/0031-9120/43/3/012

2. Simonyi K., A fizika kultúrtörténete: A kezdetektől a huszadik század végéig, Akadémiai Kiadó, 2011.

3. Juhász A., Tasnádi P., Jenei P., Illy J., Wiener C., Főzy I., A fizika tanítása a középiskolában I. http://fiztan.phd.elte.hu/letolt/fizika_tanitasa_1.pdf (Last retrieved 07/29/2023).

4. Kerettanterv a gimnáziumok 9–12. évfolyama számára. https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat/kerettanterv_gimn_9_12_evf (Last retrieved 06/29/2023).

5. Digitális kultúra 5. Oktatási Hivatal, (2020) 7–24. https://www.tankonyvkatalogus.hu/pdf/OH-DIG05TA__teljes.pdf (Last retrieved 07/12/2023).

6. Digitális kultúra 6. Oktatási Hivatal, (2021) 47-51., 54-60. https://www.tankonyvkatalogus.hu/pdf/OH-DIG06TA__teljes.pdf (Last retrieved 07/12/2023).

7. Digitális kultúra 7. Oktatási Hivatal, (2022) 54-60. https://www.tankonyvkatalogus.hu/pdf/OH-DIG07TA__teljes.pdf (Last retrieved 07/12/2023).

8. Digitális kultúra 8. Oktatási Hivatal, (2023) 24–30. https://www.tankonyvkatalogus.hu/pdf/OH-DIG08TA__teljes.pdf (Last retrieved 08/27/2023).

9. Y. Rogers, V. Shum, N. Marquardt, S. Lechelt, R. Johnson, H. Baker, M. Davies, From the BBC micro to micro:bit and beyond: a British innovation. Interactions, (2017) 24. 74–77. https://discovery.ucl.ac.uk/id/eprint/1546735/1/Rogers-Y_BBC%20Micro%20to%20micro%20paper_2017.pdf (Last retrieved 06/12/2023). https://doi.org/10.1145/3029601

10. J. Austin, H. Baker, T. Ball, J. Devine, J. Finney, P. De Halleux, S. Hodges, M. Moskal, G. Stockdale, The BBC micro:bit: from the U.K. to the world. Communications of the ACM, (2020) 63. 62–69. https://eprints.lancs.ac.uk/id/eprint/135065/1/cacmMicrobitRevisted2019_1_31.pdf (Last retrieved 07/01/2023). https://doi.org/10.1145/3368856

11. C. Andrews, BBC micro:bit - a little bit too late? [IT education]. Engineering & Technology, (2016) 11. 30–33. https://digital-library.theiet.org/content/journals/10.1049/et.2016.0400 (Last retrieved 06/01/2023). https://doi.org/10.1049/et.2016.0400

12. S. Sentance, J. Waite, S. Hodges, E. MacLeod, L. Yeomans, "Creating Cool Stuff": Pupils' Experience of the BBC micro:bit, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, ACM, Seattle Washington USA, 2017 531–

536. https://search.iczhiku.com/paper/hmF58PPakKvZrrZU.pdf (Last retrieved 08/13/2023). https://doi.org/10.1145/3017680.3017749

13. Microsoft MakeCode for micro:bit. Microsoft MakeCode for Micro:Bit,. https://makecode.microbit.org/ (Last retrieved 08/23/2023).

14. micro:bit Python Editor. https://python.microbit.org/v/3 (Last retrieved 06/13/2023).

15. Segédanyagok – micro:bit botorkálás. http://microbit.inf.elte.hu/segedanyagok/ (Last retrieved 08/12/2023).

16. A. Abonyi-Tóth, Programozzunk micro:biteket!. ELTE Informatikai Kar, (2017). http://microbit.inf.elte.hu/wp-content/uploads/2018/05/Programozzunk-microbiteket-2018.pdf (Last retrieved 08/12/2023).

17. MISZAK. http://www.inf.u-szeged.hu/miszak/ (Last retrieved 08/04/2023).

18. Gingl Z., Mellár J., Szepe T., Makan G., Mingesz R., Vadai G., Kopasz K., Universal Arduino-based experimenting system to support teaching of natural sciences. Journal of Physics: Conference Series, (2019) 1287. 012052. https://iopscience.iop.org/article/10.1088/1742-6596/1287/1/012052 (Last retrieved 08/12/2023).https://doi.org/10.1088/1742-6596/1287/1/012052

19. Makan G., Antal D., Mingesz R., Gingl Z., Mellár J., Vadai G., Kopasz K., Interdisciplinary Technical Exercises for Informatics Teacher Students. Central-European Journal of New Technologies in Research, Education and Practice, (2019) 1. 21–34. http://ojs.elte.hu/cejntrep/article/view/382 (Last retrieved 07/22/2023). https://doi.org/10.36427/CEJNTREP.1.1.382

20. Gingl Z., Makan G., Mellar J., Vadai G., Mingesz R., Phonocardiography and Photoplethysmography With Simple Arduino Setups to Support Interdisciplinary STEM Education. IEEE Access, (2019) 7. 88970–88985. https://ieeexplore.ieee.org/document/8754669/ (Last retrieved 08/04/2023). https://doi.org/10.1109/ACCESS.2019.2926519

21. Arduino alkalmazása a fizika és a digitális kultúra oktatásában | MISZAK. http://www.inf.u-szeged.hu/miszak/arduino-alkalmazasa-a-fizika-es-az-informatika-oktatasaban/ (Last retrieved 08/02/2023).

22. Piláth K., STEM receptek fizikatanároknak, ELTE Fizika Doktori Iskola, Budapest, 2021. http://fiztan.phd.elte.hu/files/kiadvanyok/STEM.pdf (Last retrieved 08/10/2023).

23. Korom E., Radnóti K., Gondolkodtató természettudomány-tanítás: Fizika | MTA-SZTE Természettudomány Tanítása Kutatócsoport, Szeged: Mozaik Kiadó, 2020. http://edu.u-szeged.hu/ttkcs/sites/default/files/konyvek/MS-9403_MTA_Fizika_online.pdf (Last retrieved 08/04/2023).

24. Teiermayer A., Improving students' skills in physics and computer science using BBC Micro:bit. Physics Education, (2019) 54. 065021. http://fiztan.phd.elte.hu/english/student/bbc.pdf (Last retrieved 08/12/2023). https://doi.org/10.1088/1361-6552/ab4561

25. R.G. Rinaldi, A. Fauzi, A complete damped harmonic oscillator using an Arduino and an Excel spreadsheet. Physics Education, (2019) 55. 015024. https://iopscience.iop.org/article/10.1088/1361-6552/ab539d/pdf (Last retrieved 08/02/2023). https://doi.org/10.1088/1361-6552/ab539d

26. D. Nichols, Arduino-Based Data Acquisition into Excel, LabVIEW, and MATLAB. The Physics Teacher, (2017) 55. 226–227. (Last retrieved 08/04/2023). https://doi.org/10.1119/1.4978720

27. Piláth K., Adatgyűjtés Excelben – Fizika kísérletek. https://pilath.wordpress.com/adatgyujtes-excelben/ (Last retrieved 08/05/2023).

28. Somogyi A., Kelemen A., Mellár J., Mingesz R., Investigation of the Rotational Motion of the Fidget Spinner by Means of Technical Informatics. Central-European Journal of New Technologies in Research, Education and Practice, (2021). (Last retrieved 03/15/2023). https://doi.org/10.36427/CEJNTREP.3.1.515

29. Somogyi A., Kelemen A., Mingesz R., Motion tracking by an Arduino Due and Excel, in: Proceedings of XXXIII. DidMatTech 2020 Conference. New Methods and Technologies in Education, Research and Practice, Eötvös Loránd University in Budapest, Trnava University in Trnava, . http://didmattech.inf.elte.hu/wp-content/uploads/2020/09/Didmattech2020_Proceedings_XXXIII_v20200921.pdf (Last retrieved 08/10/2023).

30. Kitronik Edge Connector Breakout Board for BBC micro:bit - Pre-built. Kitronik Ltd,. https://kitronik.co.uk/products/5601b-edge-connector-breakout-board-for-bbc-microbit-pre-built (Last retrieved 08/20/2023).

31. What is Rotary Encoder? Types, Principle, working in detail. Circuit Schools,. https://www.circuitschools.com/what-is-rotary-encoder-types-principle-working-in-detail/ (Last retrieved 08/20/2023).

32. Basics of Rotary Encoders: Overview and New Technologies. Machine Design, (2014). https://www.machinedesign.com/automation-iiot/sensors/article/21831757/basics-of-rotary-encoders-overview-and-new-technologies (Last retrieved 06/14/2023).

33. T.B. Greenslade, An inexpensive air track. The Physics Teacher, (1986) 24. 231–231. https://digital.kenyon.edu/cgi/viewcontent.cgi?article=1071&context=physics_publications (Last retrieved 08/10/2023). https://doi.org/10.1119/1.2341995

34. Supplementary files for the manuscript 'Event-driven kinematic measurements using BBC micro:bits programmed in C++' | MISZAK. http://www.inf.u-szeged.hu/miszak/supplementary-files-for-the-manuscript-event-driven-kinematic-measurements-using-bbc-microbits-programmed-in-c/ (Last retrieved 08/25/2023).

35. Dynamics Cart and Track System with Motion Encoder - Vernier. https://www.vernier.com/product/dynamics-cart-and-track-system-with-motion-encoder/ (Last retrieved 08/20/2023).

36. Reviews - Equipment: Pasco Smart Cart. https://iopscience.iop.org/article/10.1088/0031-9120/51/6/066001/pdf (Last retrieved 08/20/2023).

37. Hardware Overview - micro:bit V2. https://tech.microbit.org/hardware/ (Last retrieved 08/20/2023).

38. Piláth K., Micro:bit Alapú Fénykapu Építése. Fizika kísérletek, (2019). https://pilath.wordpress.com/microbit-alapu-fenykapu-epitese/ (Last retrieved 08/22/2023).

39. Micro:bitek Beszélgetnek. Fizika kísérletek, (2019). https://pilath.wordpress.com/microbitek-beszelgetnek / (Last retrieved 08/20/2023).

40. Vezetéknélküli fizika. Fizika kísérletek, (2020). https://pilath.wordpress.com/vezeteknelkuli-fizika/ (Last retrieved 08/20/2023).

41. Remote data collection. Microsoft MakeCode,. https://makecode.microbit.org/device/data-analysis/remote (Last retrieved 08/20/2023).

42. Bluetooth. Microsoft MakeCode,. https://makecode.microbit.org/reference/bluetooth (Last retrieved 08/24/2023).

43. S. Ford, T. Minshall, Invited review article: Where and how 3D printing is used in teaching and education. Additive Manufacturing, (2019) 25. 131–150. https://linkinghub.elsevier.com/retrieve/pii/S2214860417304815 (Last retrieved 06/14/2023). https://doi.org/10.1016/j.addma.2018.10.028

44. T. Ball, J. Protzenko, J. Bishop, M. Moskal, J. de Halleux, M. Braun, S. Hodges, C. Riley, Microsoft touch develop and the BBC micro:bit, in: Proceedings of the 38th International Conference on Software Engineering Companion, ACM, Austin Texas, 2016 637–640. https://dl.acm.org/doi/10.1145/2889160.2889179 (Last retrieved 08/13/2023). https://doi.org/10.1145/2889160.2889179

45. JavaScript. Microsoft MakeCode,. https://makecode.microbit.org/javascript (Last retrieved 08/20/2023).

46. Micro:bit with Arduino. Adafruit Learning System,. https://learn.adafruit.com/use-micro-bit-with-arduino/overview (Last retrieved 11/20/2022).

47. J. Devine, J. Finney, P. de Halleux, M. Moskal, T. Ball, S. Hodges, MakeCode and CODAL: Intuitive and efficient embedded systems programming for education. Journal of Systems Architecture, (2019) 98. 468–483. https://linkinghub.elsevier.com/retrieve/pii/S1383762118306088 (Last retrieved 11/13/2022). https://doi.org/10.1016/j.sysarc.2019.05.005

48. The micro:bit runtime DAL/CODAL. https://tech.microbit.org/software/runtime/ (Last retrieved 08/20/2023).

49. micro:bit (nRF51, Cortex-M0) GPIO toggling. https://metebalci.com/blog/microbit-gpio-toggling/ (Last retrieved 08/20/2023).

50. CODAL target for the micro:bit v2.x series of devices. (2022). https://github.com/lancaster-university/codal-microbit-v2 (Last retrieved 08/20/2023).

51. CODAL build tools and sample programs for the micro:bit v2.x. https://github.com/lancaster-university/microbit-v2-samples (Last retrieved 08/20/2023).

52. radio - micro:bit runtime. https://lancaster-university.github.io/microbit-docs/ubit/radio/ (Last retrieved 08/20/2023).

53. serial - micro:bit runtime. https://lancaster-university.github.io/microbit-docs/ubit/serial/ (Last retrieved 08/20/2023).

54. UARTService - micro:bit runtime. https://lancaster-university.github.io/microbit-docs/ble/uart-service/ (Last retrieved 08/26/2023).

55. Library for MicroBitUARTService. GitHub Page of Lancaster University,. https://github.com/lancaster-university/codal-microbit-v2/blob/master/inc/bluetooth/MicroBitUARTService.h (Last retrieved 08/26/2023).

56. How to convert an integer into a specific byte array in C++. Educative: Interactive Courses for Software Developers,. https://www.educative.io/answers/how-to-convert-an-integer-into-a-specific-byte-array-in-cpp (Last retrieved 08/26/2023).

57. Microsoft Data Streamer Add-in for Excel - Excel Data Streamer. https://learn.microsoft.com/en-us/microsoft-365/education/data-streamer/ (Last retrieved 08/20/2023).

58. John-Hart, Excel solutions - Visual Studio (Windows). (2023). https://learn.microsoft.com/en-us/visualstudio/vsto/excel-solutions?view=vs-2022 (Last retrieved 08/29/2023).

59. John-Hart, Ribbon XML - Visual Studio (Windows). (2023). https://learn.microsoft.com/en-us/visualstudio/vsto/ribbon-xml?view=vs-2022 (Last retrieved 08/29/2023).

60. Archiveddocs, Customizing the 2007 Office Fluent Ribbon for Developers (Part 1 of 3). (2014). https://learn.microsoft.com/en-us/previous-versions/office/developer/office-2007/aa338202(v=office.12) (Last retrieved 08/29/2023).

61. Archiveddocs, Customizing the 2007 Office Fluent Ribbon for Developers (Part 2 of 3). (2014). https://learn.microsoft.com/en-us/previous-versions/office/developer/office-2007/aa338199(v=office.12 (Last retrieved 08/29/2023).

62. Archiveddocs, Customizing the 2007 Office Fluent Ribbon for Developers (Part 3 of 3). (2014). https://learn.microsoft.com/en-us/previous-versions/office/developer/office-2007/aa722523(v=office.12) (Last retrieved 08/29/2023).

63. Karl-Bridge-Microsoft, Bluetooth Low Energy in Universal Windows Platform apps - UWP applications. (2023). https://learn.microsoft.com/en-us/windows/uwp/devices-sensors/bluetooth-low-energy-overview (Last retrieved 08/26/2023).

64. BBC micro:bit Bluetooth Profile - micro:bit runtime. https://lancaster-university.github.io/microbit-docs/ble/profile/#introduction (Last retrieved 08/26/2023).

65. Karl-Bridge-Microsoft, Bluetooth GATT Client - UWP applications. (2023). https://learn.microsoft.com/en-us/windows/uwp/devices-sensors/gatt-client (Last retrieved 08/26/2023).

66. VSPT Overview/Virtual Serial Port Tools Online Documentation. https://hhdsoftwaredocs.online/vspt/virtual-serial-port-tools-documentation/overview.html (Last retrieved 08/26/2023).

67. P. Peterlin, Data analysis and graphing in an introductory physics laboratory: spreadsheet versus statistics suite. European Journal of Physics, (2010) 31. 919–931. (Last retrieved 07/28/2023). https://doi.org/10.1088/0143-0807/31/4/021

68. Guidelines and examples of array formulas. https://support.microsoft.com/en-us/office/guidelines-and-examples-of-array-formulas-7d94a64e-3ff3-4686-9372-ecfd5caa57c7 (Last retrieved 08/15/2023).

69. Create a named range from selected cells in a worksheet - Microsoft Support. https://support.microsoft.com/en-us/office/create-a-named-range-from-selected-cells-in-a-worksheet-fd8905ed-1130-4cca-9bb0-ad02b7e594fd (Last retrieved 08/29/2023).

70. Insert a named range into a formula in Excel - Microsoft Support. https://support.microsoft.com/en-us/office/insert-a-named-range-into-a-formula-in-excel-131f2e66-30b6-4f1a-afb3-83007f63082d (Last retrieved 08/29/2023).

71. A.M. Brown, A step-by-step guide to non-linear regression analysis of experimental data using a Microsoft Excel spreadsheet. Computer Methods and Programs in Biomedicine, (2001) 65. 191–200. https://doi.org/10.1016/S0169-2607(00)00124-3

Authors

SOMOGYI Anikó
University of Szeged, Faculty of Science and Informatics, Department of Technical Informatics;
Radnóti Miklós Experimental High School of Szeged, Hungary,
e-mail: somogyia@inf.u-szeged.hu

KELEMEN András
University of Szeged, Juhász Gyula Faculty of Education, Department of Applied Information Technology;
Radnóti Miklós Experimental High School of Szeged, Hungary,
e-mail: kelemen@jgypk.szte.hu

MELLÁR János
University of Szeged, Faculty of Science and Informatics, Department of Technical Informatics, Hungary,
e-mail: mellar@inf.u-szeged.hu

MINGESZ Róbert
University of Szeged, Faculty of Science and Informatics, Department of Technical Informatics, Hungary,
e-mail: mingesz@inf.u-szeged.hu

About this document

Published in:

CENTRAL-EUROPEAN JOURNAL OF NEW TECHNOLOGIES IN RESEARCH, EDUCATION AND PRACTICE

Volume 6, Number 1. 2024.

ISSN: 2676-9425 (online)

DOI:

10.36427/CEJNTREP.6.1.7135

License