**Session 2**

**Latest Library PiZumo1B**

In this session we are going to use one of the LED lights which is attached to our microcontroller and the small LCD screen. We will introduce some further programming concepts in Python and at the end you will have programmed a one blinking light and the LCD to write a message.

**LED light**

Here is a diagram of the Zumo robot.



The LED lights are behind the USB programming interface near the buttons A, B and C. There is a yellow, red and green light. The red and green lights show that information is travelling between the Pi and the Zumo, so we will only program the yellow light.

Pololu (the manufacturers of the Zumo robot) have developed a library of functions which control all parts of the robot. These commands are in the language C so CDU lecturers have written two libraries, one in the Zumo and one in the Pi so that we can call these libraries from Python. Make sure that you are using the latest version of the library, PiZumov1A. If not see your lecturer. The libraries that are written at CDU are updated at the end of each semester.

Now open up the Geany and create a new file called blinktest.py. Don’t forget to create a folder for this session. Type the following:

#! /usr/bin/env python3

from PiZumov1B import Zumo

from time import sleep

zumo = Zumo()

sleep(1)

while True:

 zumo.ledYellow(1)

 sleep(1)

 zumo.ledYellow(0)

 sleep(1)

**This is a lot of code and if you try to run in now it will not work. You also need to find the file called PiZumov1B and make a copy inside the folder where you saved this file. When you run it you will now see the yellow light should blink on and off. You will also see the red light blinking, this is showing that the information is being sent from the Pi to the Zumo.**

**from PiZumov1B import Zumo**: **import** is a Python command which fetches functions and objects from other files, in this case the PyZumov1 file. This command looks inside the PiZumov1B.py file and gets the object called Zumo and makes it available to us. Objects are beyond the scope of this course so we won’t go into how to create them and how they work, that is done in the Cert IV. We will simply show you how to use them. The object Zumo that we have fetched is how we communicate with the zumo robot. We give this object the commands it requires and it then sends these commands to the zumo robot. PiZumov1B is a simple Python file so you could look inside it if you wish to see how it works, however be careful not to change anything if you want it work.

**from time import sleep:** this command is very similar, however it get a function called sleep from the standard library called time. The Standard Libraries come with Python so we don’t need to place a copy in our folder.

**zumo = Zumo():** here we are creating a variable call zumo. The value we are giving is the object that has been imported. The () runs this object which allows it to set up the connection between the Raspberry Pi and the Zumo through the serial cable.

**sleep(1)** is the Library function from the **time** library that we imported and it will stop the execution of the program for the number of seconds that we give it. We give the zumo robot one second to complete all the changes it needs to start accepting our commands from the Pi.

**while True:** **while** is a Python key word which creates a loop. This command takes an expression which must be true or false. If the expression is true is will run the block of code attached to it. If it is false it will force the program to drop down beyond the block of code. In Python **if** statements and **while** loops work the same except that **if** statements only run once while **while** loops will run each time the expression is true. **True** is a Python key word which is always true. The effect of this statement is that it creates an endless loop.

**ledYellow** – this is an zumo robot function that applies turns the yellow light on and off depending on the argument passed to it, 1 turns the light on, and 0 turns the light off. The PiZumo library will only accept a 1 or a 0, if you try anything else the program will crash and you will get an error message. You can look in the library to see how this was done if you like.

All that code to simply turn a light on and off. Image how much code you would need to write if you wanted something more complicated, there must be a better way. There will be a lot of occasions when we want to turn a light on on our vehicle, we don’t want to write all that code every time.

The better way is to create our own library with functions in it that we can use. Make the following changes:

def blink():

 zumo.ledYellow(1)

 sleep(1)

 zumo.ledYellow(0)

 sleep(1)

while True:

 blink()

We have now created a new function called blink and simply used it in the endless loop. We have not gained much yet.

This is better, but currently blink only works with one time setting, it would be better to make it much more general, we do that by adding parameters. Change blink to the following:

def blink(timeon=1, timeoff=1):

 zumo.ledYellow(1)

 sleep(1 \* timeon)

 zumo.ledYellow(0)

 sleep(1 \* timeoff)

**def blink(timeon=1, timeoff=1):** this introduces a new way of doing parameters (if you can’t remember what parameters are look back at the last Session). Blink has two parameters (local variables used inside the function) but this time they have been given default values. This means it is up to the user to determine whether to keep the default value (by just leaving out the argument when the function is called) or to change them. After making this change our program still works the same.

We now have the problem, how do we know what values to use with the function? There are two ways to solve this problem, one is to use good variable names like here. The other is to add a comment so you can remember and other people can read them to find out how the function works.

**Comments**

Comments are added to code by programmers to explain to other programmers what is happening. The best comments explain what is supposed to happen, not how things work. You can expect anyone reading you code to understand the code. However while you are learning you might want to add comments to your code to help you remember how things work.

There are two types of comments in Python, a **#** makes a single line comment. Anything from then on in the line is a comment. Eg

print("Hello")#This prints the word on the screen

In this code everything from the **#** onwards is ignored by Python. You can put these in front of a line to cause that line to not work, this is useful in testing where we can add print lines to work out what is going on and then take them out later.

The other type of comment is the multiline comment which starts with a triple quote and ends with a triple quote.

One place where you should always make a comment is when you create functions, use a triple quote to explain what the function is trying to do. This comment can be placed either directly above the function or as the first line. In our main PyZumo library there is a comment right at the start of the library with a lot of information as well as comments for every function.

So let’s add a comment to our function to explain how it works. Add the following

def blink(timeon=1, timeoff=1):

 '''The variables timeon and timeoff have been set to one second as default'''

 zumo.ledYellow(1)

 sleep(1 \* timeon)

 zumo.ledYellow(0)

 sleep(1 \* timeoff)

This uses the triple quotes to create a multiline comment which describes the way the function works.

Now we need to write a couple of tests to make sure that this function works properly. Replace the endless loop with the following tests

print("Testing the yellow light")

print("This will blink the light for one second")

blink()

print("This will blink the light 4 times quickly")

blink(.5)

blink(timeon=.5)

blink(.5)

blink(timeon=.5)

print("This will blink the light for 5 seconds")

blink(5)

print("Tests over")

This is now almost finished. It still has couple of problems which will show up when we try to use it. However all the basics are there for a good library.

1. It has a useful general function that can be used in many programs
2. It has everything it needs inside the file
3. It has tests and comments to make sure that it works properly

Save this file as blink.py and then create a new file called main.py, make sure they are saved in the same folder. In this new file write the following code:

#! /usr/bin/env python3

from blink import blink

while True:

 blink()

You should understand this code line by line and know exactly what it is going to do. Run it and you will see an unexpected and unwanted result. You not only get the blinking light that we expected (eventually), but you also get the tests that we wrote for blink. We don’t want those test when we are using a library, only when we are testing it. This is one of the problems I mentioned earlier and there is a solution.

In your blink file we will put the tests behind a special **if** statement like so:

if \_\_name\_\_=='\_\_main\_\_':

 print("This will blink the light for one second")

 blink()

 sleep(1) ...

 …

Now run the main.py file and you should just get the blinking light, no test. Run the blink.py file and you still get the tests.

**if \_\_name\_\_=='\_\_main\_\_':** This is Python’s way of working out which file is running the code and which file is a library. These tests are in the block so will only run when the file is the run. When the file is imported these tests do not run. For a detailed answer see this stack overflow item <http://stackoverflow.com/questions/419163/what-does-if-name-main-do> .

Now blink is finished and can be used as a library whenever you need a blinking light, which we will need for the rest of the course. There are problems still with this, see if you can work out what they are and fix our blink library.

*The Zumo actually has three lights: yellow, red and green. The yellow light works very well, while the red and the green are used by the Zumo to show data transfer. Despite this the green light also works well from the library while the red light does work at times. You could use these light if you wish using library commands.*

**LCD Screen**

There are two functions in our PiZumo library that control the LCD screen. They are:

lcdClear and lcdPrint

lcdClear has no parameters and will clear the LCD screen of all its contents

lcdPrint has one parameter and will print only the first 8 characters, which is all that can fit on a small screen.

Create a new file and type in the following:

#! /usr/bin/env python3

from PiZumov1B import Zumo

from time import sleep

zumo = Zumo()

sleep(1)

while True:

 zumo.lcdClear()

 zumo.lcdPrint("Hello")

 sleep(1)

 zumo.lcdClear()

 zumo.lcdPrint("there")

 sleep(1)

**Glossary**

|  |  |
| --- | --- |
| Concepts | Meaning |
| Command | These are key words used by the language that perform a function for the language. It is also possible to create your own commands to be used in your programming. Most languages have libraries of commands that have already been built for you to use as well as those you build yourself. |
| Functions | The most common way to create your own commands. These can accept data, process data in some way and can return the changed data to be used in other parts of the program. In a well structured program almost everything is done in small single purpose functions. |
| Arguments | These are bits of data that are used by functions, so that they have information to work with. |
| Parameters | These are the place holder variables that are used in function definitions. These are replaced by the arguments that are used when the function is run. Often the two terms are used interchangeably.In Python Parameters can be given a default value, which means that an argument is not required when the function is called. |
| String | Data type, String refers to ordinary words.  |
| Integers | Data type, int refers to whole numbers.  |
| Float | Data type, Float used for decimals, but they are not accurate.  |
| Variable | Way of representing data for the program to work on. |
| Libraries | These contain commands that have been developed and tested and are ready to use. Many of the libraries have been written by the people who originally developed the language while other libraries have been developed by companies or individuals that use the language (Google, Yahoo, Apache etc have all developed extensive language libraries for a wide variety of languages). You can also develop your own library of commands. |
| Assignment | Uses = to give a variable name a value |
| Expression | An operation which must be true or false |
| Comments | These are notes for people reading the code. |
|  |  |
| **Python Commands** | Meaning |
| print | Python function which will show the data on the console |
| def | Python key word used to define your own functions |
| input | Python function which gets data from the keyboard |
| int | Python function which changes a String to its Integer value |
| if | Python keyword used to build decision making structures, must be followed by an expression which is either true or false |
| from … import | Python keywords used to bring in Python objects and function from other files for use in your current file. |
| while | Python keywords which creates a loop based upon the expression |
| True | Python keyword which always evaluates to true, there is also a keyword False. |