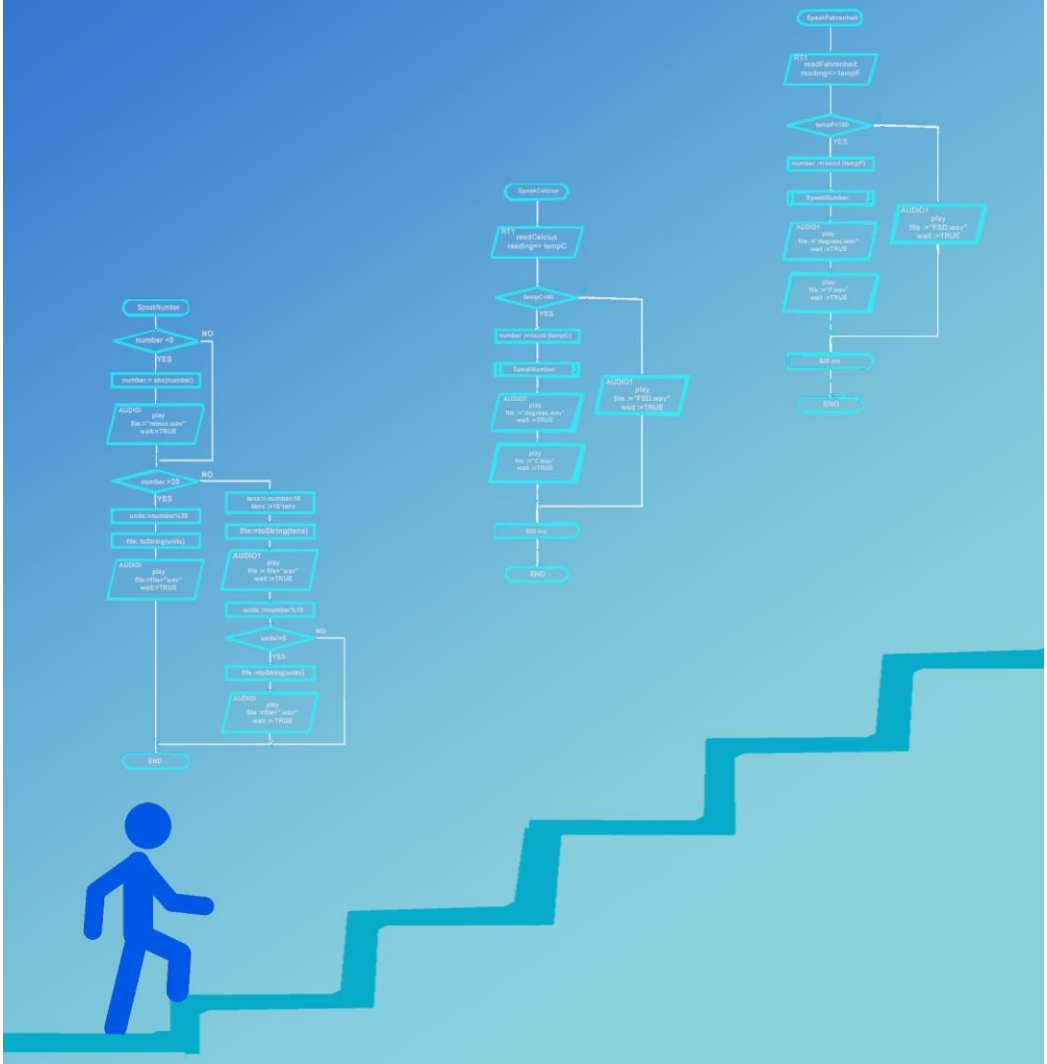
# LEARN PROGRAMMING AND ELECTRONICS WITH PROTEUS VISUAL DESIGNER



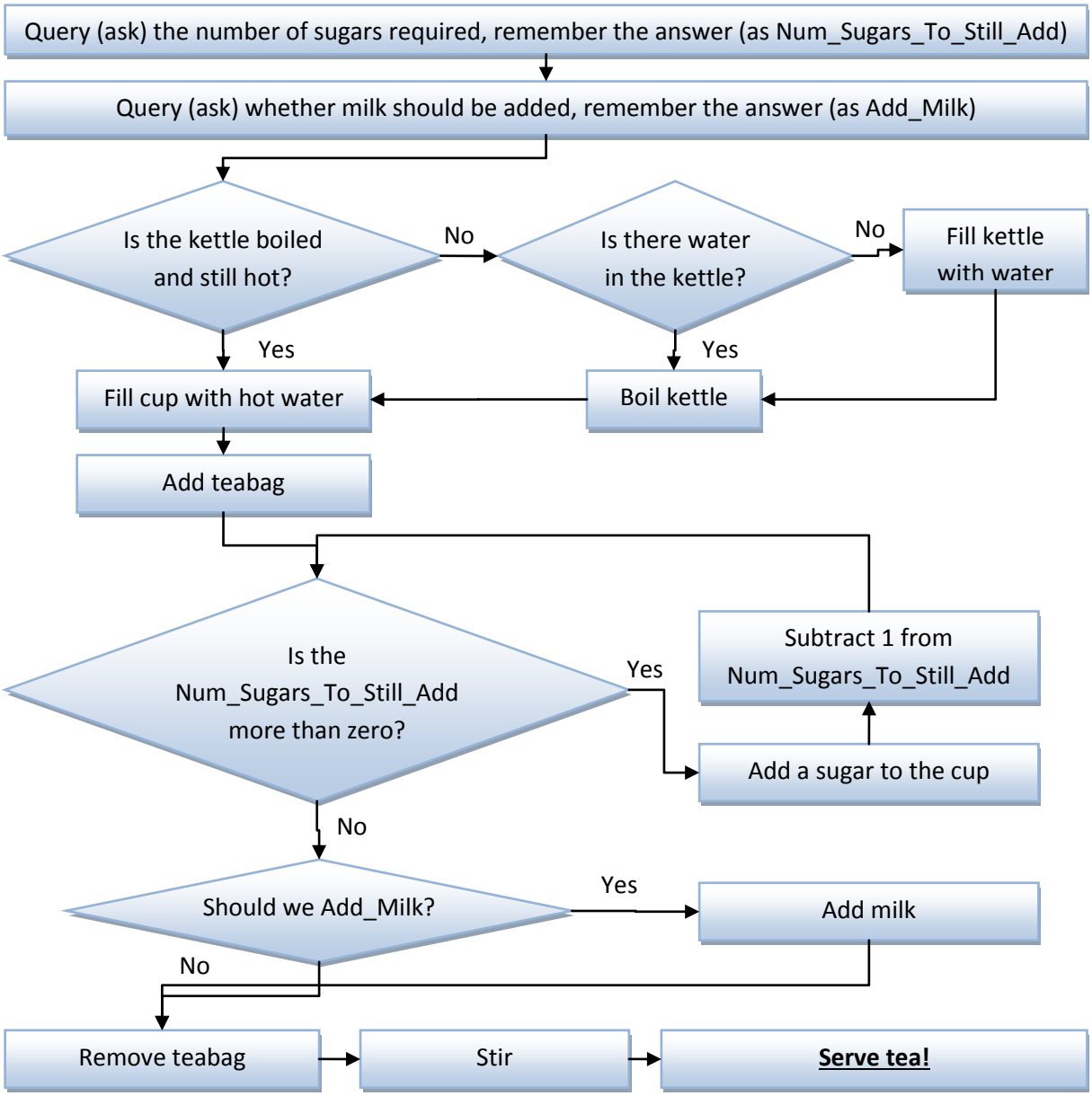

PROTEUS

Labcenter Electronics www.labcenter.com

## 1.2. Terms and Technologies

### 1.2.1. Flowchart

A flowchart is a visual, easy-to-understand diagram which illustrates a sequence of decisions and actions required to perform a task. For example, let's look at a program which describes how to make a cup of tea:




Notice the procedural ("step-by-step") structure of the flowchart, as well as the binary ("yes or no") decision making process – this is the way in which a computer "thinks".

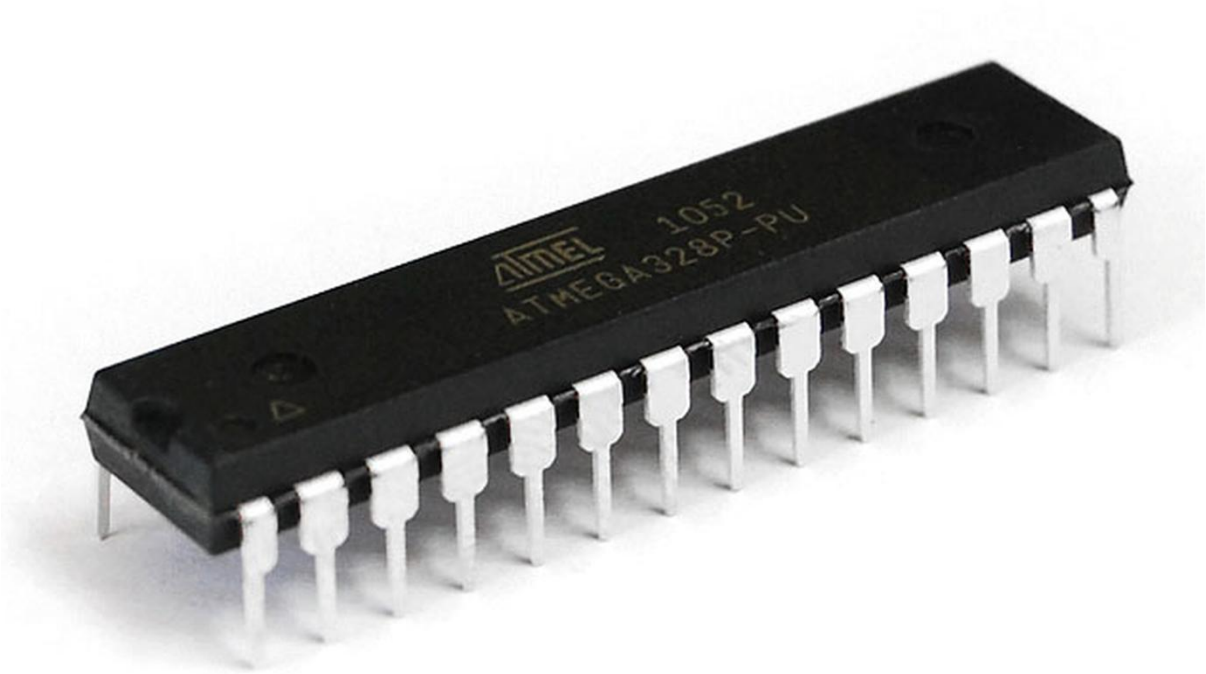
### 1.2.2. "C" Code

"C" is an industry-standard "high-level" programming language, used for writing software programs. "Industry standard" means that the technology is very popular and common in the software industry. "High-level" means that the code is more like English and less like raw computer-code. A tool called a "compiler" converts the high-level language into low-level computer-code. For beginners, C can seem confusing and difficult at first. This book focuses first on flowchart programming, which is easier to understand and less prone to errors such as "typing mistakes", and then introduces C towards the end of the book. Note that even professional C programmers will still use flowcharts from time to time, even if just on paper; to plan out a program, or to provide a summary of how a program works for a report.

An even higher-level version of C, called "C++", adds some more complex functionality to the C "language".

Note that all of the flowcharts in this book can be converted into C/C++ code with a few clicks (the exact procedure for doing this will be explained towards the end of the book, in section 9.8).

### 1.2.3. Microcontroller




A microcontroller is an electronic chip which can process instructions (run software), and also has input/output connections ("pins" or "legs") which enable it to interact with the "outside world". Microcontrollers also generally include some "extra" features (called "peripherals"), such as timers, which can help them to accomplish more than they could by just processing code only. You may have heard of the term "CPU" or "microprocessor", which is the "brain" inside a computer that processes instructions; a microcontroller is a microprocessor, but with extra features (peripherals) built-in.

### 1.2.4. Arduino


Arduino is an extremely popular and open-source series of low-cost electronic circuit boards ("open-source" means that the design "source" files are freely available to the public). The boards are designed to make it quick and easy to build electronic machines – by providing a ready-made platform on which to build – and feature a microcontroller as well as pins for inputs and outputs.

The term "Arduino" technically also refers to the Arduino "IDE[2]" and compiler; these are computer programs in which software code for the Arduino boards can be written (in C/C++). Since we will be programming Arduino boards using flowcharts first, we will not worry about the Arduino IDE / compiler or C/C++ code for now (these topics are introduced towards the end of the book, in chapter 9).

[2] IDE stands for "Integrated Development Environment", and is largely a document editor with some special features which assist in writing computer code – such as highlighting elements of the code in different colours.

# 2. Creating Your First Project

To get started, we are going to jump right in and see how easy it is to create our first working project!

## 2.1. Pre-Requisites

- This chapter assumes that you already have some basic computer knowledge and experience (that you know how to use a computer).
- Please ensure that Proteus is installed on your computer before proceeding.

## 2.2. Creating a New Project

**Step 1:** To start with, please ensure that Proteus is running (this can normally be done by double-clicking on the Proteus icon on your Windows desktop). When Proteus starts, the Proteus Home Page is shown, as illustrated in the following figure.

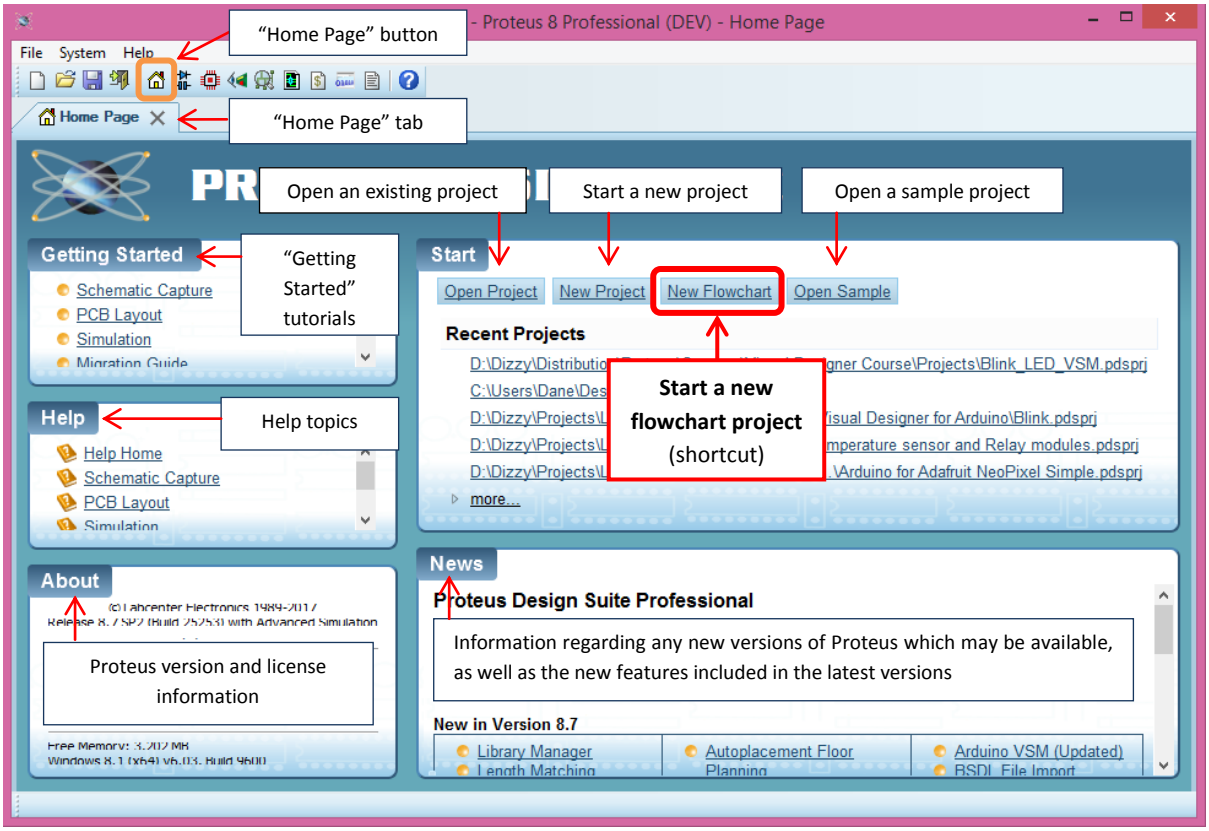



Figure 2-1: Proteus "Home Page"

**Step 2:** In order to start a new flowchart project, the **"New Flowchart"** option should be selected in the **Start section** of the Proteus desktop screen (see Figure 2-1). This will then launch the New Project Wizard for starting a new flowchart project (Figure 2-2).

(Tip: The "New Flowchart" option is similar to the more advanced "New Project" option, but streamlines the process by leaving out (or using default values) for steps which are not usually used in a flowchart project. Proteus has other capabilities besides for flowchart design, but we will not be looking at these just yet.)

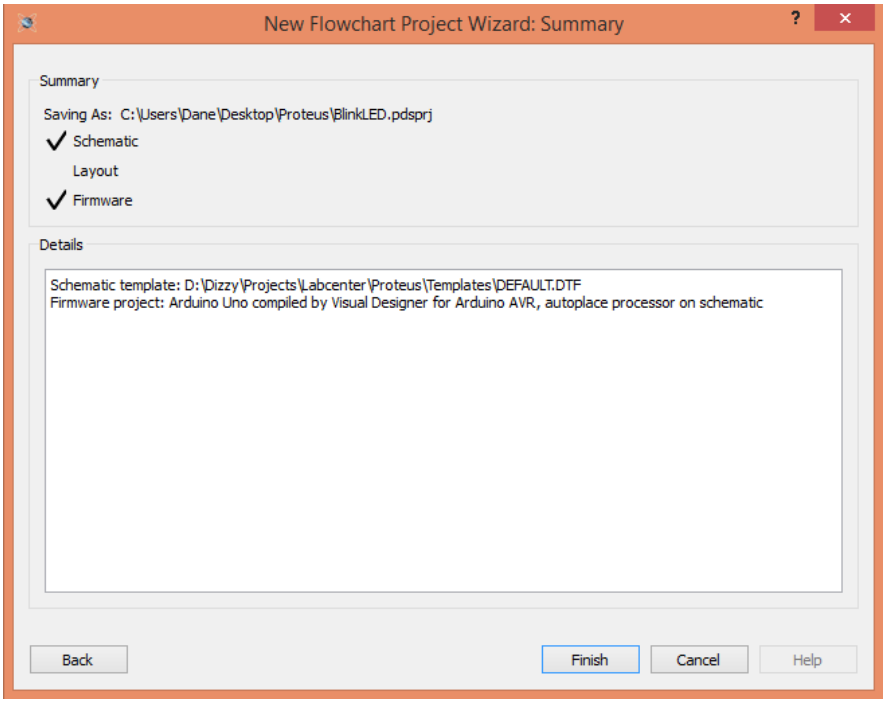



Figure 2-4: Summary of details for new project

Click the **"Finish"** button at the bottom of the screen to finalise the wizard process. After clicking the Finish button, Proteus will automatically open the newly created project (Figure 2-5). Once you have reached this stage, you have successfully created a Proteus project and can commence flowchart design.

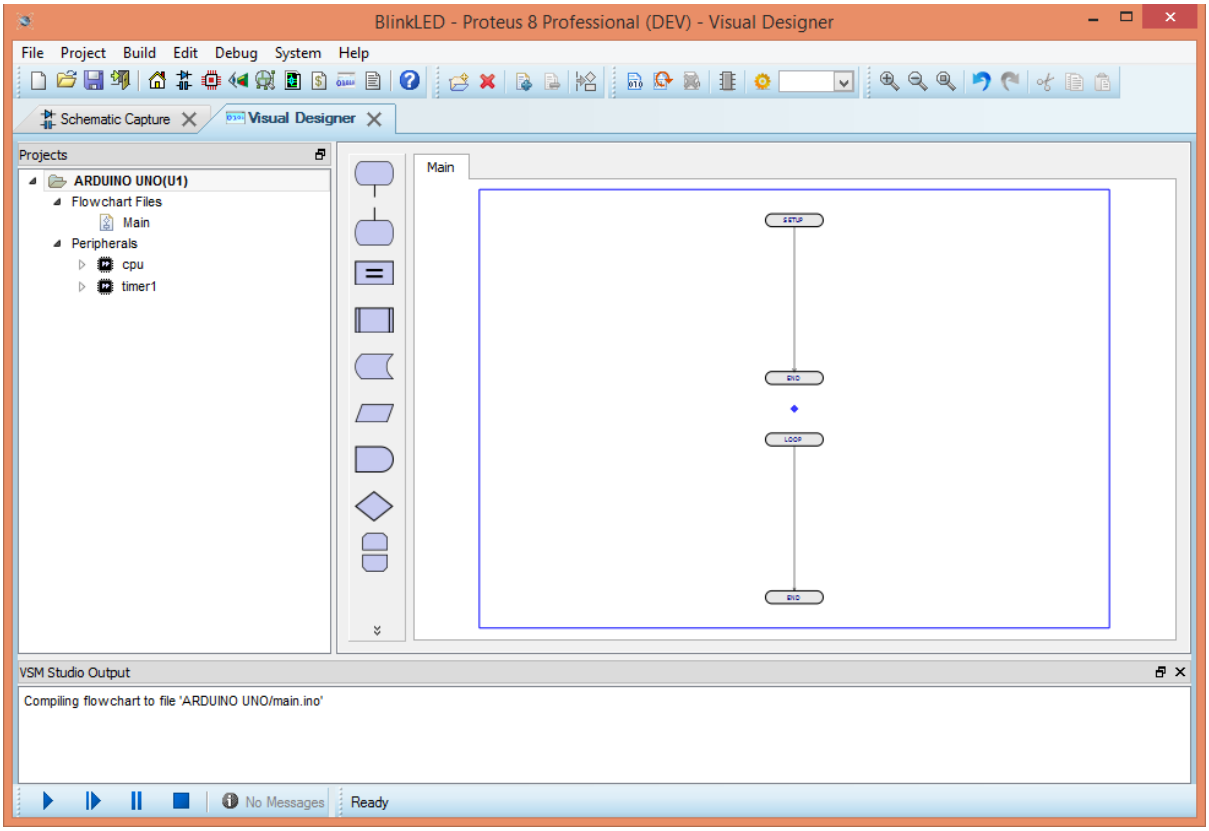



Figure 2-5: New flowchart project

## 2.3. Familiarisation with the Proteus Environment

After creating your new project using the Wizard process, Proteus will have two tabs open – the Visual Designer tab and the Schematic Capture tab – with the Visual Designer tab selected by default.

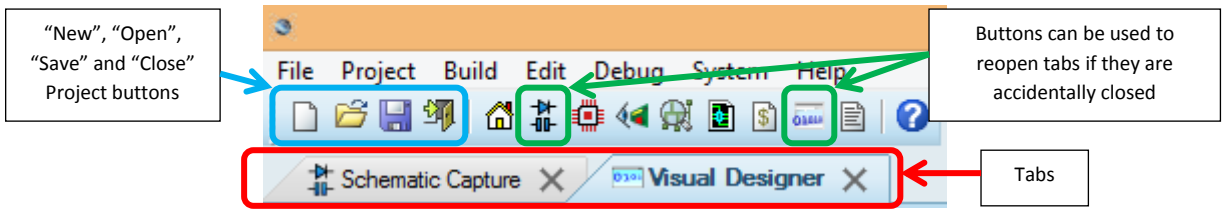



Figure 2-6: Schematic Capture and Visual Designer tabs

### 2.3.1. Schematic Capture Tab

The Schematic Capture tab contains the electronic circuitry for the project. **We can safely ignore this tab for now**, as the Wizard process automatically places the necessary circuitry for our project onto the schematic. If you're curious, then you can take a quick peek to see what it looks like.

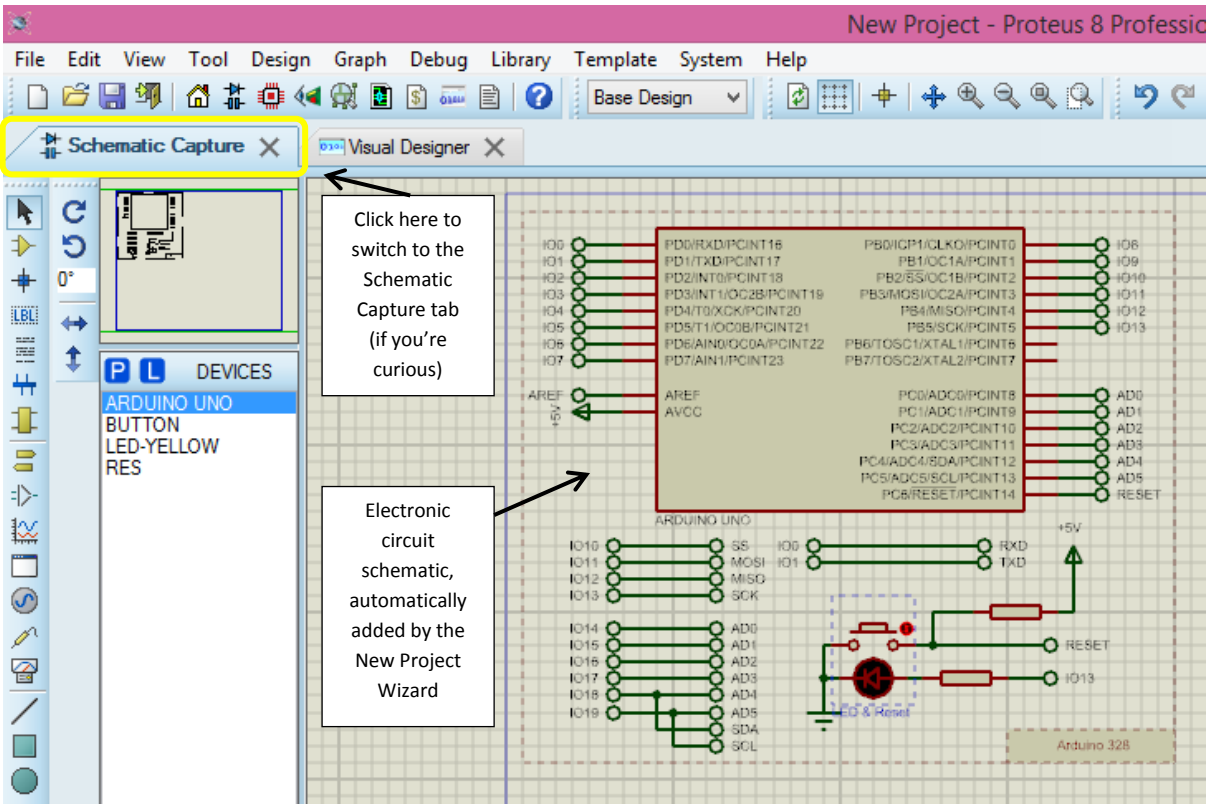
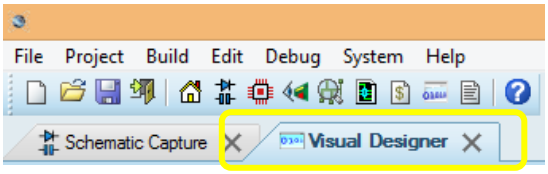



Figure 2-7: Schematic Capture tab

(Tip: For a reminder of what a schematic is, please see section 1.2.5 in the Introduction.)

Once done, switch back to the Visual Designer tab.



### 2.3.2. Visual Designer Tab

The Visual Designer tab is where flowchart programs are designed.

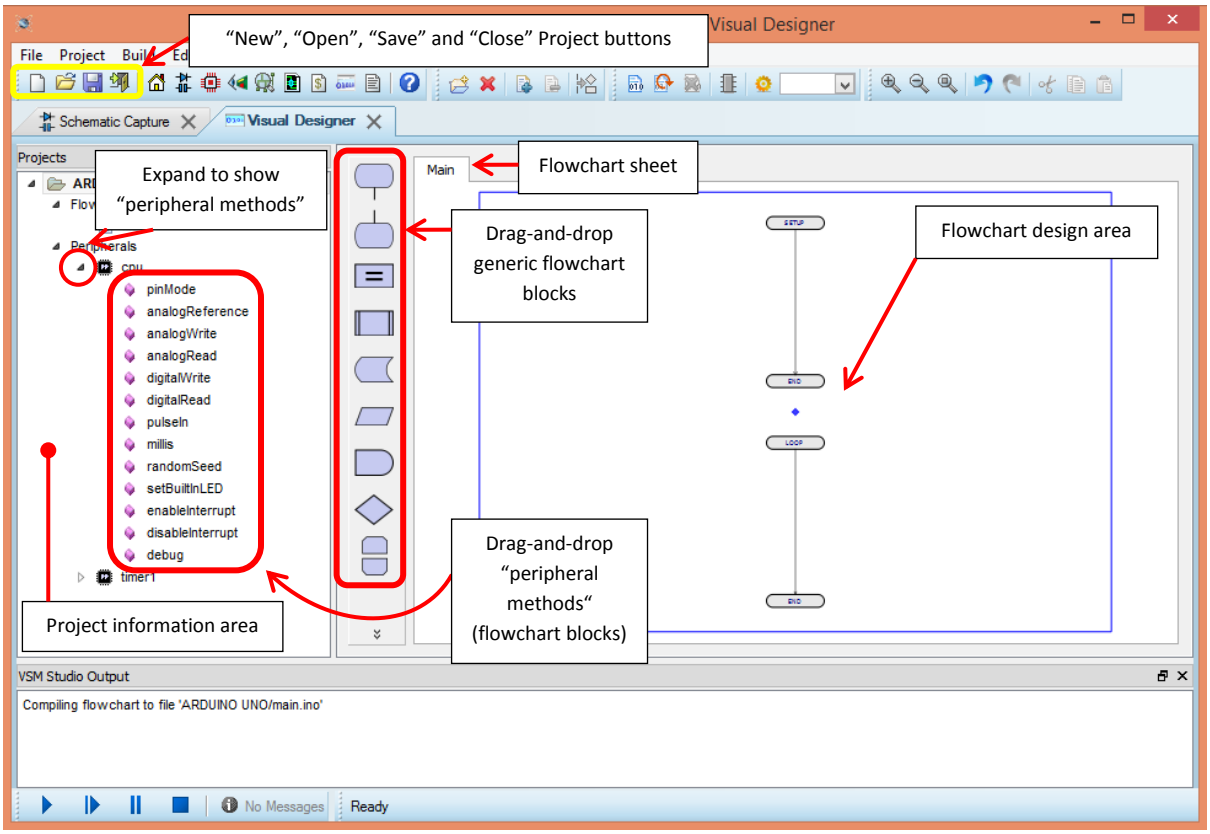



Figure 2-8: Visual Designer tab

When a new flowchart project is created then the flowchart area contains by default a "Setup" and a "Loop" routine (these will be explained in greater detail shortly). The small "Projects" area on the left lists all of the flowchart sheets in the project, as well as the "peripherals" used in the project. More information on these concepts is presented next.

### 2.3.3. Flowchart Sheets

A flowchart sheet is a "page" on which you can design a flowchart. For many projects only one flowchart sheet is needed, but for larger projects more flowchart sheets can be added in order to provide more space for designing, or to better organise the project.

### 2.3.4. Peripherals

In short, in Visual Designer, peripherals are largely objects which we can interact with – for example a button or a display screen. Peripherals have "methods" (instructions), such as "clear the screen" or "display this text on the screen". The longer, and more technically correct, explanation is as follows:

As mentioned in the introduction (section 1.2.3), a microcontroller is a microprocessor with some extra features ("peripherals") built-in. Whilst a microprocessor can only process instructions, *peripherals* provide additional functionality – such as a timer peripheral for (obviously) timing things. The peripherals work independently from the microprocessor, and as such don't slow it down from processing instructions. This is like having a GPS in your car – you can focus on driving, whilst the

Other options include the below, although these generally only work within the schematic capture tab:

- To simply “pan” the Editing Window up, down, left or right; position the mouse pointer over the desired part of the design and press the F5 key.
- Hold the SHIFT key down and bump the mouse against the edges of the design area to pan up, down, left or right.
- Use the Pan Icon on the toolbar.

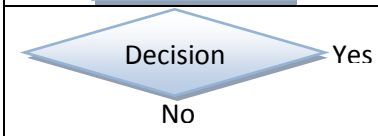
Take some time now to practice zooming and panning around the flowchart sheet (and possibly the schematic sheet as well, if you feel so inclined).

## 2.4. Essentials of Flowchart Programming

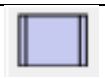
A flowchart consists of a sequence of actions and decisions required to accomplish a task, drawn in a graphical way. For an example, see the flowchart from section 1.2.1 in the Introduction again. A flowchart is generally easier to understand than written (text) code (e.g. C code). By using flowcharts for programming we can focus on getting results, without first having to tackle the complexity of learning for example C or C++ code.

### 2.4.1. Types of Flowchart Blocks

Flowcharts are constructed from “blocks”. Traditionally, flowcharts are built out of only action and decision blocks:

| Block | Name | Description |
|---|---|---|
|  | Action Block | Represents an action to be taken, for example switch on a light. |
|  | Decision Block | Represents a decision which should be taken, with different possible outcomes. |

Proteus however has a richer set of flowchart blocks to choose from, which offer some additional features; as listed in the following table. **Don’t worry about understanding or remembering all of the blocks right now** – they will be covered in greater detail in the following sections, and are listed here only for the sake of completeness.

| Functional block | Name | Description |
|---|---|---|
|  | Event Block | The event block, together with the end block, is used to create a sub-routine (covered in Chapter 8). The event block can also be used to handle events such as interrupts, however this is currently beyond the scope of this book[5]. |
|  | End Block | The end block is used to signify the end of a sub-routine or event. |
|  | Assignment Block | The assignment block is used to assign a value to a variable in the program. Variables are used whenever something needs to be remembered, and will be described in greater detail later. In the "Make Tea" example from the introduction (section 1.2.1), "Num_Sugars_To_Still_Add" and "Add_Milk" are variables. |
|  | Sub-routine Call Block | A "Sub-Routine" in Visual Designer is a group of flowchart blocks which are placed separately to the main flowchart. Grouping certain tasks into sub-routines can help to make the flowchart more organised and easier to understand, and can also prevent duplication of "code". Sub-routines are covered in Chapter 8. A sub-routine is executed (or "run") by using a Sub-routine Call Block. |
|  | Stored data block | The stored data block is used to perform actions (i.e. reading, writing) on storage objects (e.g. SD cards). |
|  | I/O[6] (Peripheral) Operation Block | This block is used to represent a peripheral method (see section 2.3.4), such as switching a light on or off or writing some text to a display. |
|  | Time Delay Block | This block creates a time delay in the program. For example, "Wait 5 seconds before moving onto the next task". |
|  | Decision Block | Represents a decision which should be taken, with different possible outcomes, such as "Is the button pressed?" (yes or no). |
|  | Loop Construct Blocks | Loops can also be created using decision blocks, but the Loop Construct Blocks make it easier to, for example, repeat a task a certain number of times. |
|  | Interconnector | If a flowchart starts to become too large, then interconnectors can be used to break the flowchart up into separate sheets or sections (joined by the interconnectors). |
|  | Comment Text | This "block" can be used to add some descriptive text or notes to your flowchart, but does not affect the operation of the program in any way. |

[5] Suggestions for further learning on this subject are made in section 12.1.

[6] "I/O" stands for "Input / Output". Reading the temperature from a thermometer would be an example of an input, and playing a song via an audio speaker would be an example of an output.

### 2.4.2. Flowlines

Flowchart blocks are linked, or joined, by lines called “Flowlines” (see Figure 2-9).

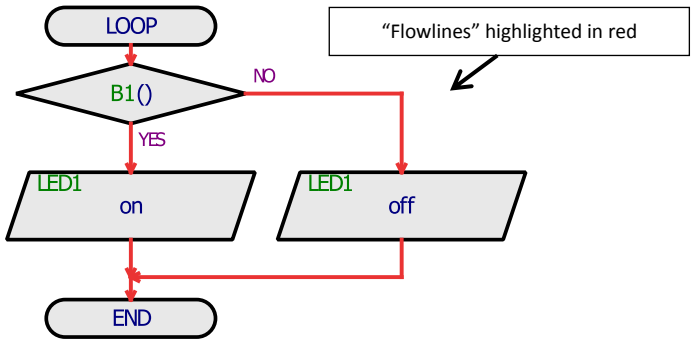



Figure 2-9: Flowlines join flowchart blocks

### 2.4.3. Setup and Loop and Routines

When a new Proteus Visual Designer project is created, there are by default two routines included on the flowchart: “SETUP” and “LOOP”.

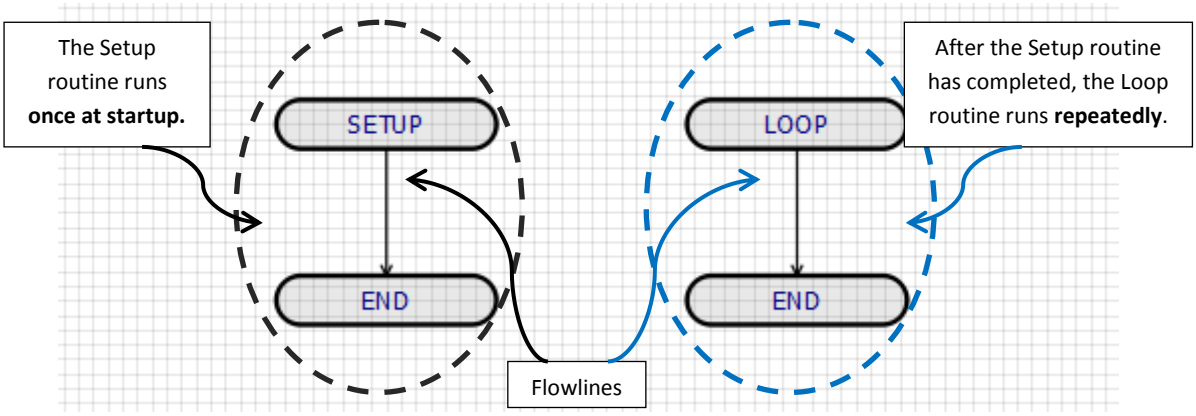



Figure 2-10: Default flowchart program with setup and loop routines

The **Setup** routine is executed (or “run”) **once** at initial start-up.

After the Setup routine, The main **Loop** routine is executed (or “run”) **repeatedly**. Most of the program is normally constructed within the Loop routine.

## 2.5. Designing a Flowchart to Blink an LED

We will now proceed to design a flowchart which turns an “LED” on and off. LED stands for “Light Emitting Diode”, and is essentially a small light. LEDs are extremely common, and you will have encountered them many times in everyday life (for example, many electronics device have a small LED light which indicates whether they are powered on or not).

For this project we will toggle (switch) the LED on and off, with a 1 second delay in-between. This is a popular task when starting any new project; it is a program which can be relatively quickly developed, and serves to prove that the system (microcontroller and electronics) is “working”, which is normally done as a test before moving onto developing more complex programs.

Details regarding exactly what LED we will be blinking are presented in section 2.7.1, but for now we will jump straight into designing the flowchart program (please feel free to read section 2.7.1 before continuing, if you are curious).

### 2.5.1. Drawing the Flowchart

The flowchart which we will be building looks like the following figure:

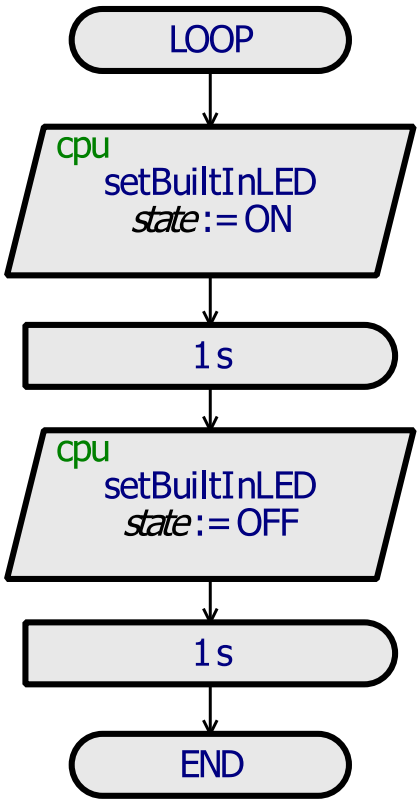



Figure 2-11: Blink LED flowchart

To construct the flowchart, please complete the following instructions:

**Step 1:** Drag a “setBuiltInLED” method from the CPU peripheral into the flowchart **Loop** routine (Figure 2-12). The method is named “setBuiltInLED” because it can be used to turn a built-in LED on the Arduino Uno board on or off (this is discussed in section 2.7.1 – feel free to read that section and then come back if you are curious).

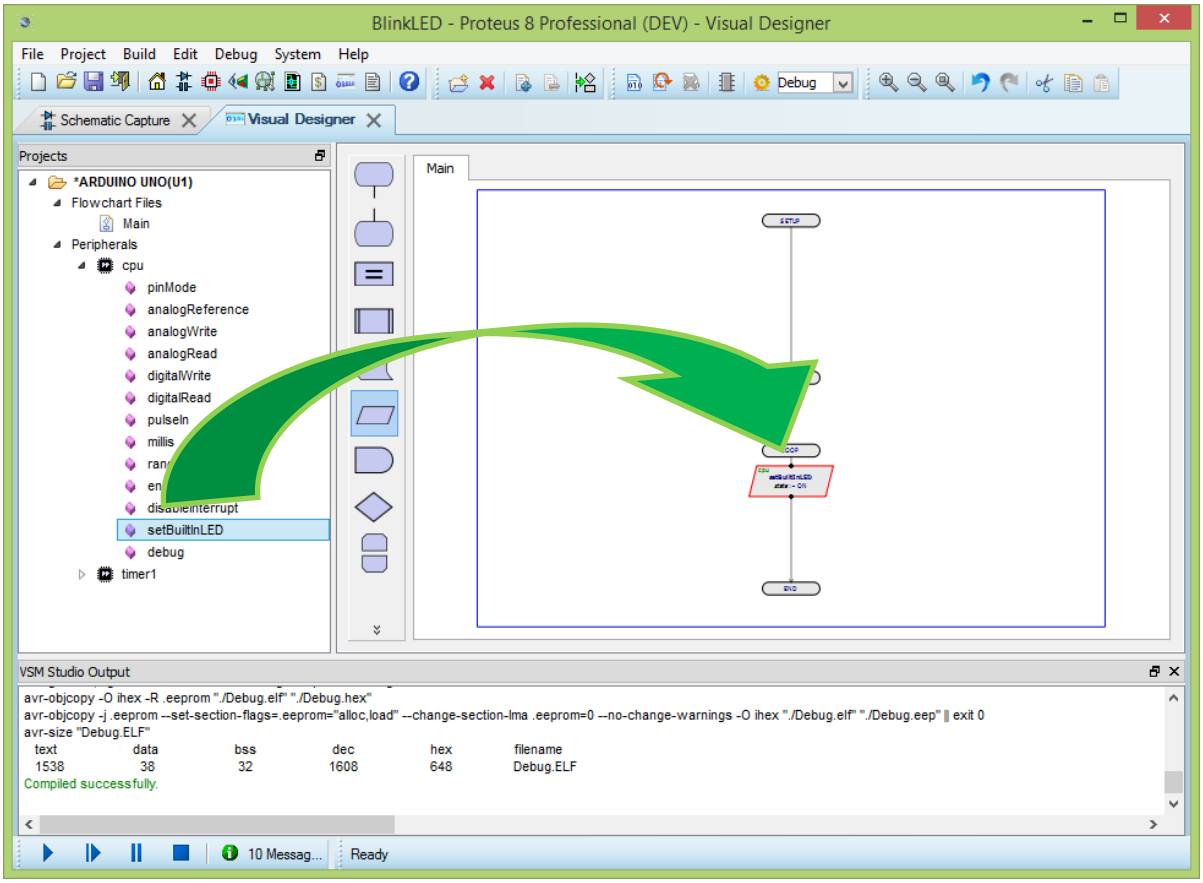



Figure 2-12: Adding the "setBuiltInLED" method to the flowchart

When correctly aligned, black dots will appear where the block intersects the flowline (Figure 2-13).

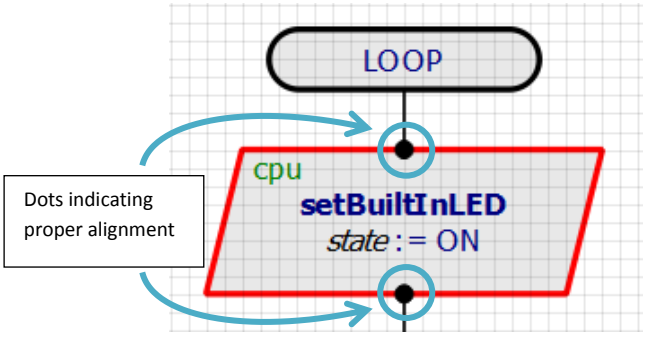



Figure 2-13: Inserting a flowchart block

Once properly inserted, there should be a flowline coming in and a flowline going out of the block, with no dots displayed anymore (Figure 2-14), and arrows at the end of the flowlines.

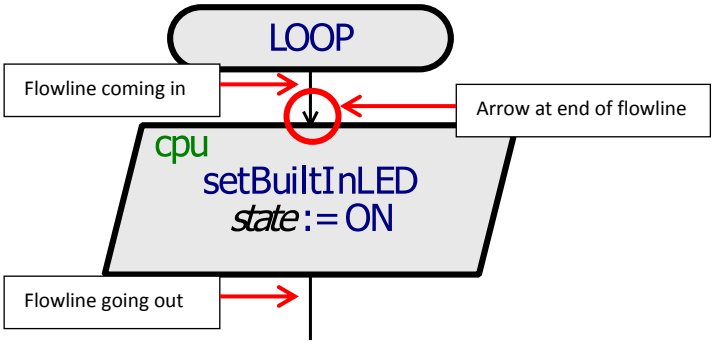



Figure 2-14: Inserted flowchart block

By default, the “setBuiltInLED” method will switch the LED “on” (“state := ON”). This is the correct operation for now, so we do not need to change it. How to change a setBuildInLED block so that it switches the LED off again will be covered in steps 4 and 5.

**Step 2:** The next step is to add a delay, before we switch the LED off again. This is done using a delay block; so drag a delay block into the flowchart, inserting it after the setBuiltInLED block (Figure 2-15).

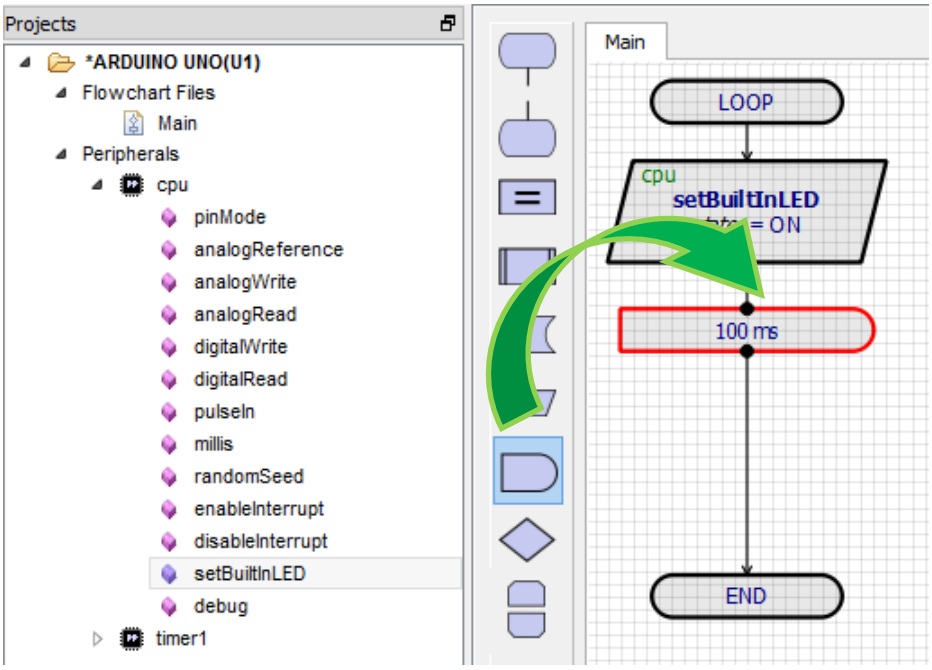



Figure 2-15: Inserting a delay block

**Step 3:** Next, we want to set the time of the delay to be 1 second. Right-click on the new delay block which was just added to the flowchart, and select "Edit" from the popup menu (Figure 2-16). An alternative method is to left-click on the delay block twice.

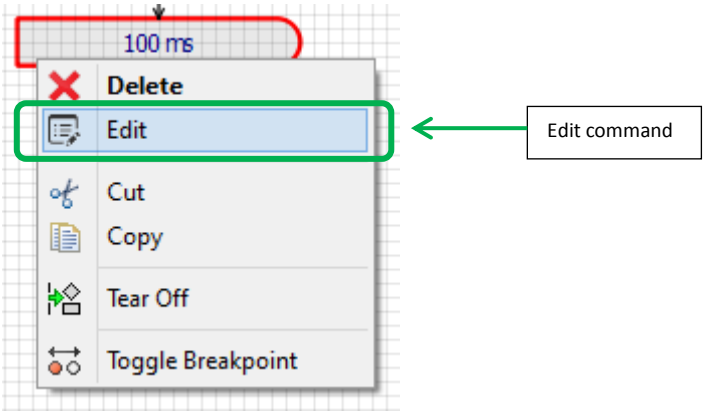



Figure 2-16: Delay block popup menu

Specify "1 second" for the delay (Figure 2-17), and click "OK" when done.

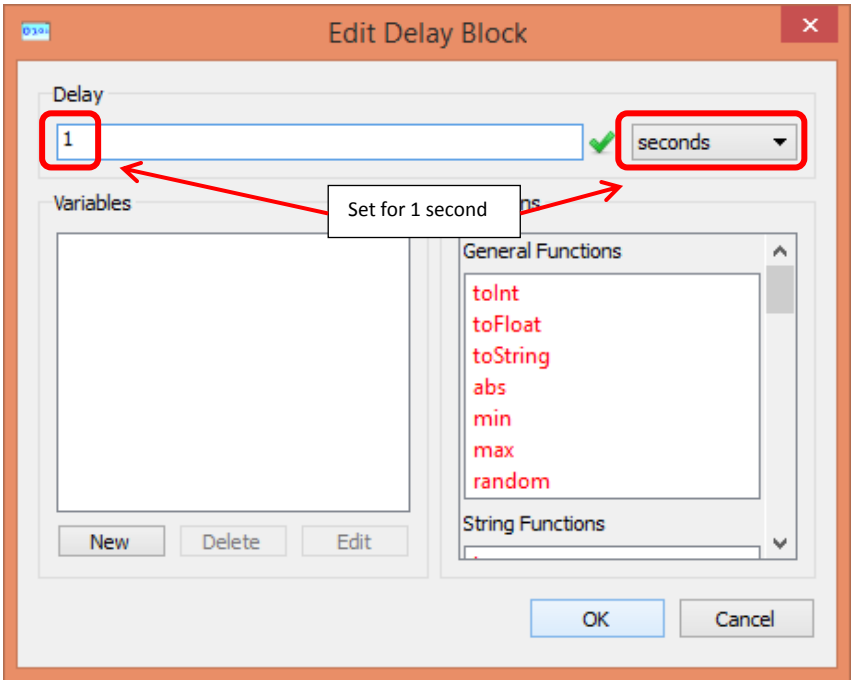



Figure 2-17: Specifying the delay time

The delay block should now look like Figure 2-18.


Figure 2-18: Delay block set to 1 second

**Step 4:** After switching the LED on for 1 second, we now need to switch it off again for another 1 second.
In order to switch the LED off again we need another setBuiltInLED block – so drag another setBuiltInLED method from the cpu peripheral into the flowchart (similar to Step 1), inserting it after the 1s delay.

By default, the setBuiltInLED method switches the LED “on”; we need to change our newly added block so that it instead switches the LED “off”. In order to do so, right click on the block and select “Edit” from the popup menu.

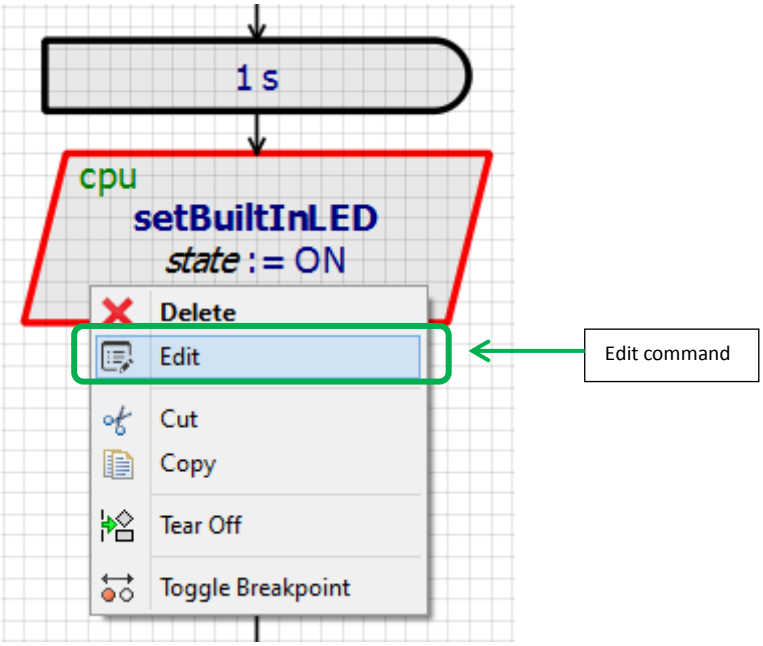



Figure 2-19: setBuiltInLED block popup menu

The “Edit I/O Block” Dialog is now displayed (Figure 2-20).

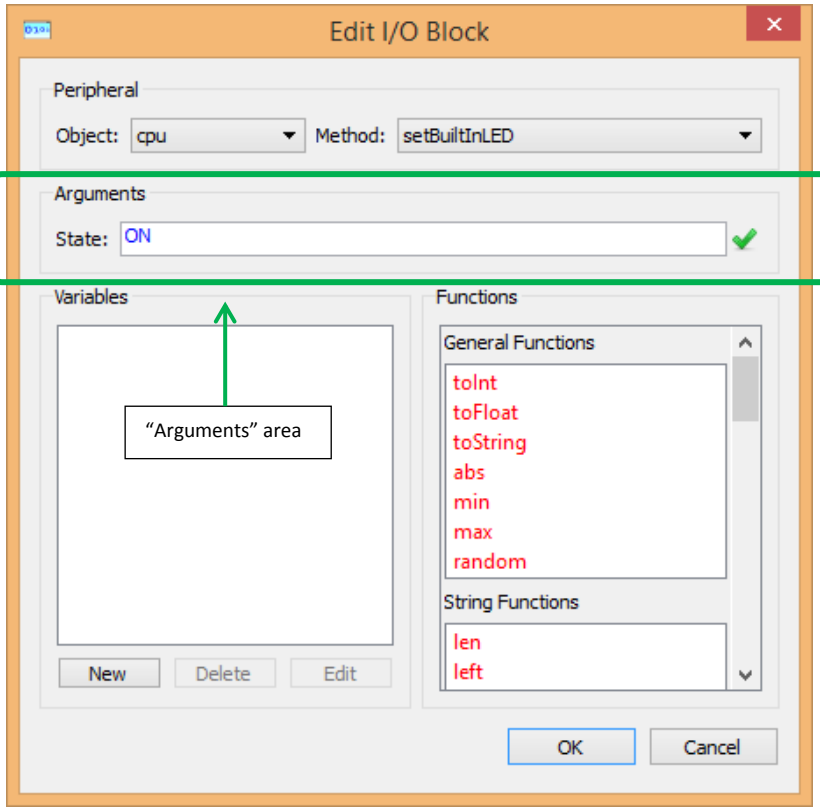



Figure 2-20: Edit I/O Block dialog

**Step 5:** In the Edit I/O Block dialog, enter "OFF" for the "State:" argument.

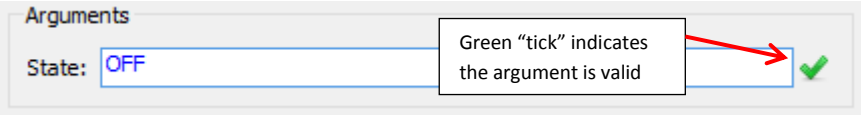



Figure 2-21: Specifying the "State" argument

In programming, an **argument** means a piece of information passed to a method (also sometimes called a "parameter"), to give it more information regarding what it should do. In this case we have the setBuiltInLED method which can turn the LED on or off, but we have to specify which of the two options we want. Note that the term "argument" in programming has **nothing to do with the normal sense of the word** ("diasagreement")!

Once ready click "OK". The setBuiltInLED flowchart block should now look like the following figure:


Figure 2-22: setBuiltInLED routine with "state" set to "OFF"

(Tip: For readers who have some prior programming experience, you may be interested to know that "ON" in Proteus Visual Designer is a defined alias for "TRUE".)

**Step 6:** Now that the LED has been switched off again, we need to keep it off for 1 second before the loop repeats and switches the LED back on again. In order to do so, add a 1s delay after the second setBuiltInLED routine (similar to steps 2 and 3).

Once done, the Loop routine of the flowchart should look like the following figure.

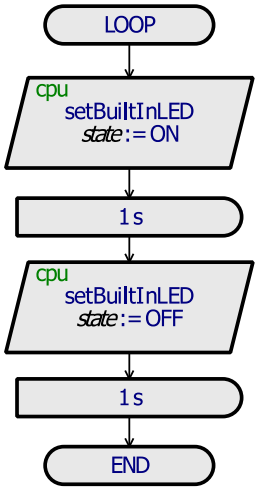



Figure 2-23: Blink LED flowchart

Check that the sequence and details of the blocks in your flowchart exactly match those of Figure 2-23 (if not, then please go through the steps of this tutorial again to check that nothing has been missed). The flowchart Setup routine should still be empty.

Congratulations! The “Blink LED” program flowchart is now complete.

## 2.6. Running the Simulation

To see our flowchart program in action, we can proceed to running the simulation. To do so, click the blue “Play” button at the bottom-left of the Proteus window (Figure 2-24).

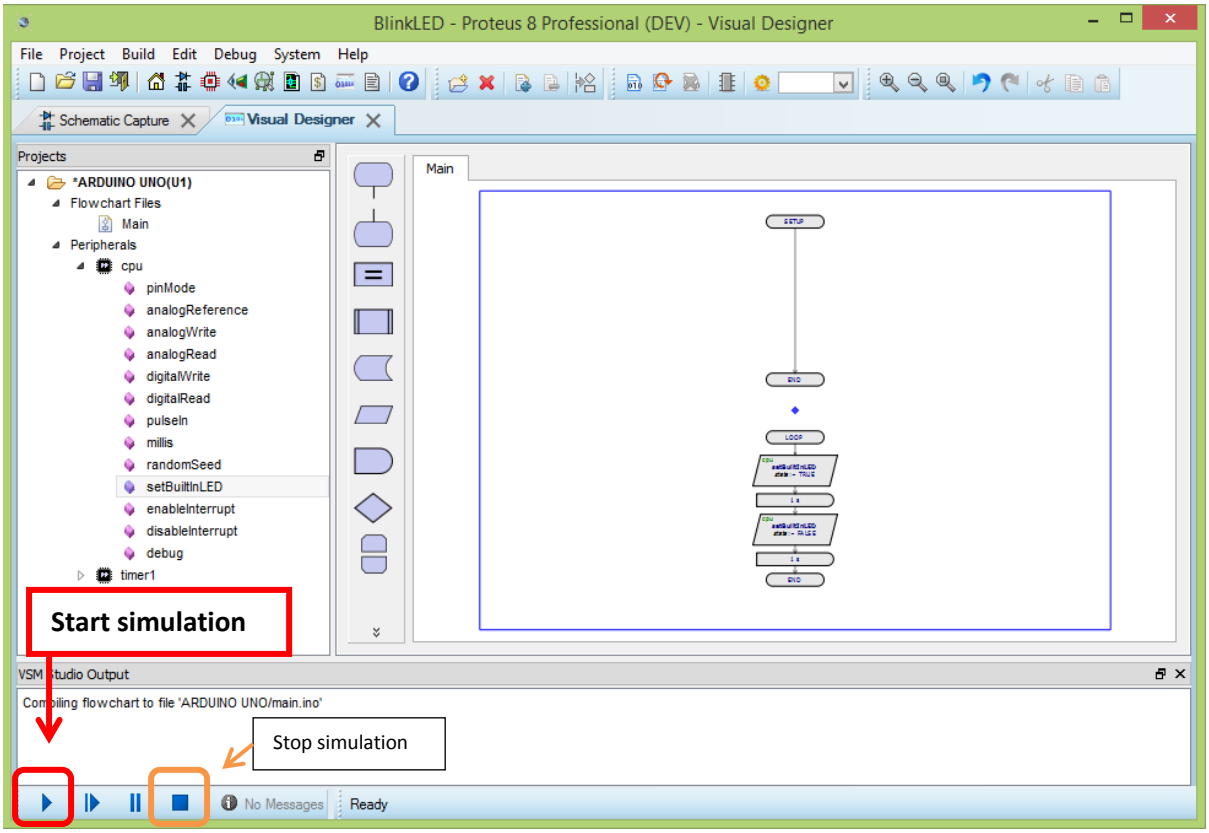



Figure 2-24: Starting the simulation

Upon clicking the blue triangular play button, Proteus first compiles the flowchart so that it can be simulated (please see Step 3 of section 2.2 if you need a reminder of what a compiler does).

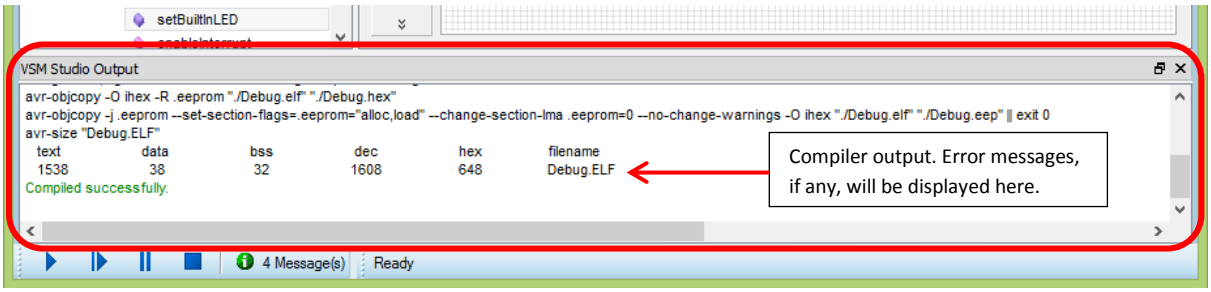



Figure 2-25: Compiler output

If there are any problems with the flowchart, error messages will be displayed in the compiler output. If not, then a green “Completed successfully” message will be displayed.

(Tip: Compilation can also be triggered manually by selecting “Build Project” from the “Build” menu.)

Once the project has compiled successfully, the simulation will start (Figure 2-26).

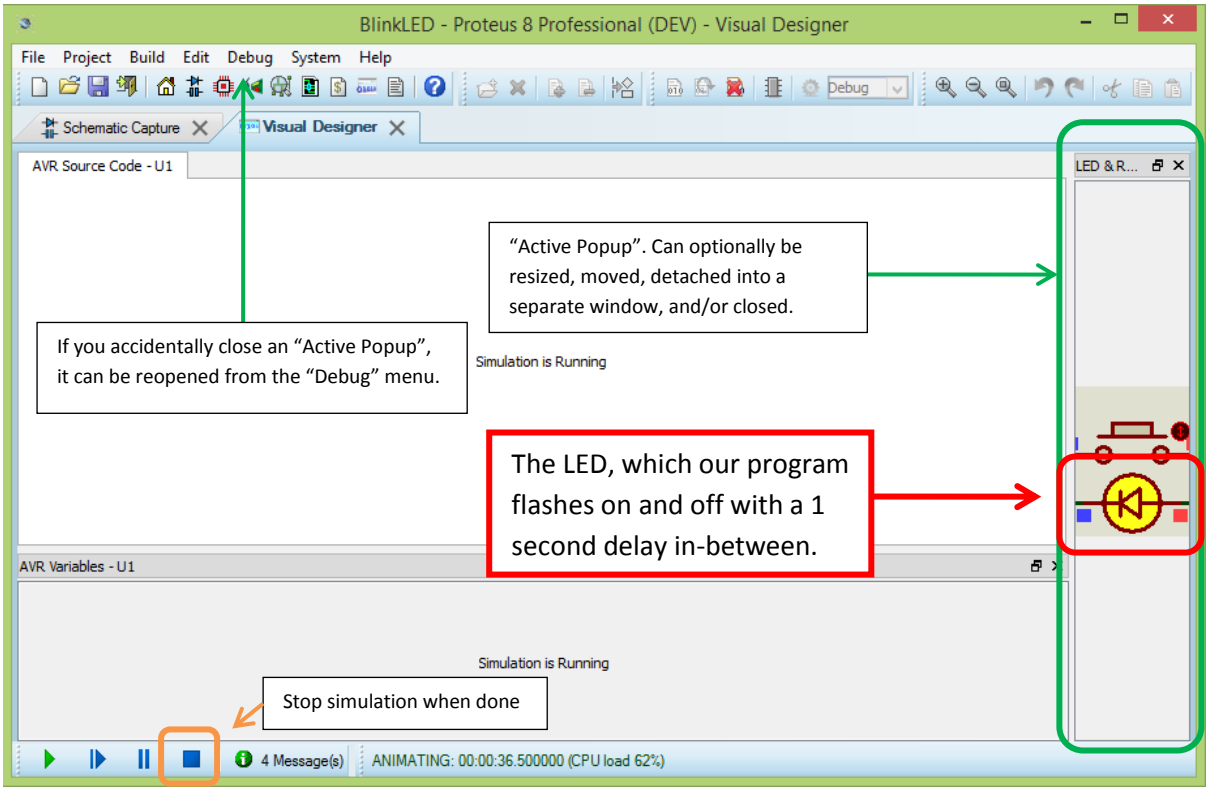
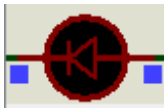



Figure 2-26: Running simulation

You should see a yellow LED flashing on and off with a 1 second delay in-between, as specified by our flowchart program.

LED On:  LED Off: 

Once done, click the blue rectangular "Stop" button (see Figure 2-26) to end the simulation. Congratulations! You have now built and simulated a working program.

(Tip: You may wish to save your work. This can be done using the "Save Project" command available from the File menu, by clicking on the Save button ( ) on the toolbar, or by pressing Ctrl + S on your keyboard.)

If you would like to experiment a little more, then try changing the time of the delays and see how this changes the timing of the LED blinking.

If you are wondering how your program can be transferred to "real world" hardware, then please note that this is discussed in section 2.7.4.

## 2.7. Further Information

### 2.7.1. What LED Are We Blinking?

The Arduino Uno which we selected for use in our project during the "New Project Wizard" (see Figure 2-3) is a low-cost "development board"; it essentially features a microcontroller, some

supporting electronic circuitry (such as circuitry for handling the power supply to the board), and a built-in on-board LED. Connectors ("headers" or "sockets") are also featured around the edges of the board, which can be used to more easily connect external peripheral circuits to the microcontroller (see Figure 2-27).

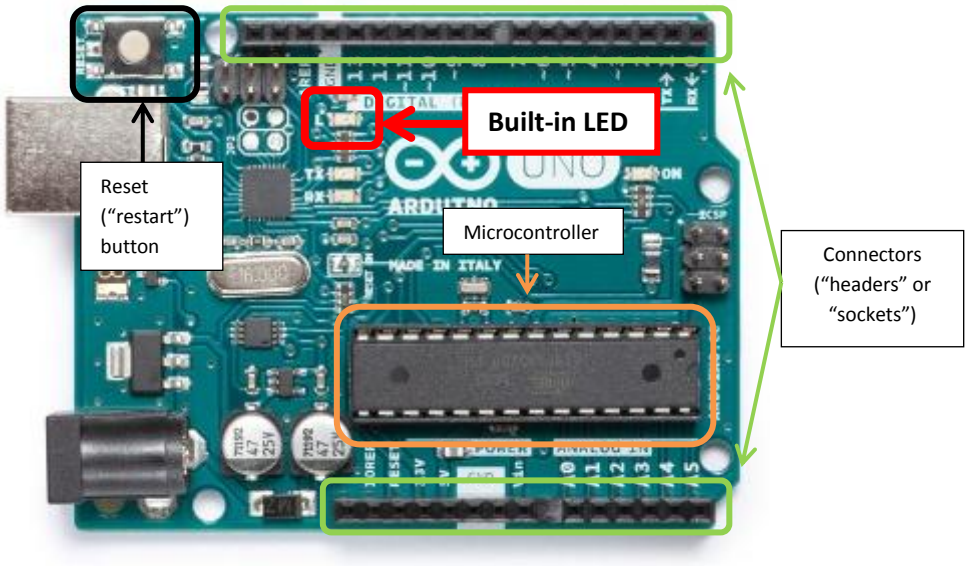



Figure 2-27: Arduino Uno board[7]

In Proteus, the Arduino Uno circuitry is contained in the Schematic Capture tab. As mentioned in section 2.3.1, **it is not necessary to understand the schematic capture tab or circuitry for now**, however for the curious the schematic circuit looks as follows:

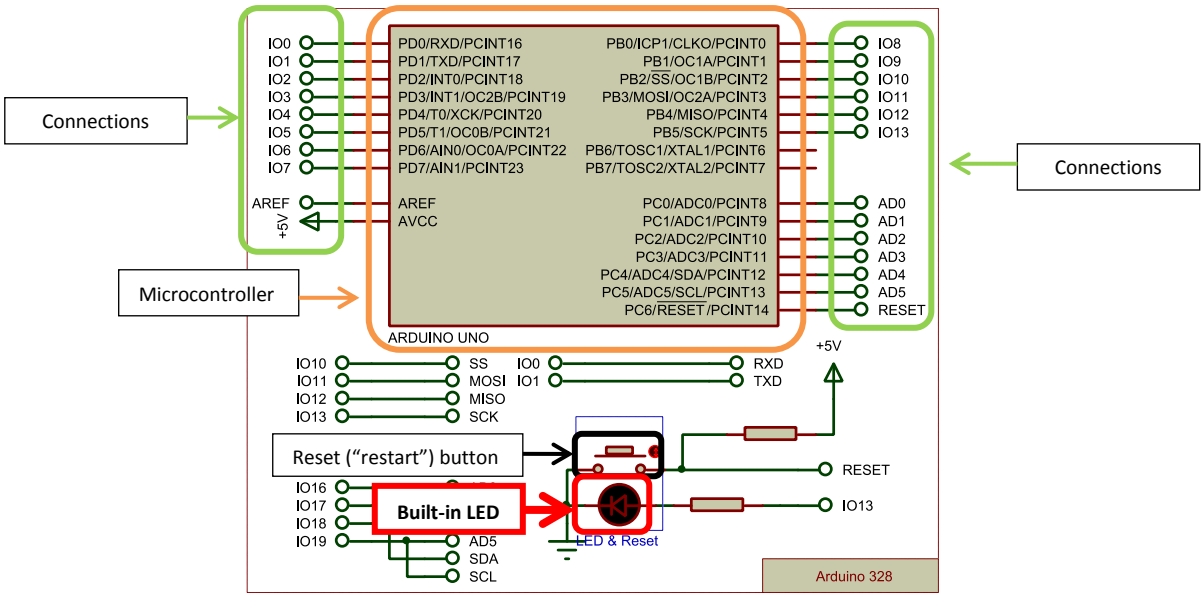



Figure 2-28: Arduino Uno schematic

[7] Image from https://store.arduino.cc/usa/arduino-uno-rev3