Firefly

Users manual

I n d e x

1. Overview

Setting up the remote control system of your observatory requires quite a bit of effort – our Firefly has been designed to help, making it easier.

The Firefly is a very straightforward device, easy to learn and to setup. The only effort required is to understand the way relays are used (if you are not yet familiar with relays). We'll learn to setup and use the Firefly with a working example.

So, first of all, a bit of background.

A relay is a very useful device indeed – being an electrically operated switch, it enables appliances such us our Firefly to turn on and off other appliances.

Let's compare relays and push buttons.

Every relay has two sides: the control side (your finger able to push or release the button) and the switch properly said (the internals of the pushbutton opening or closing the circuit).

The most common push buttons leave the circuit open if nobody is pushing them, and close the circuit when pushed (there also exist push buttons working the other way around). These most common push buttons are called "normally open": they are open until some effort is made to change the state. The other model, yes, is called "normally closed".

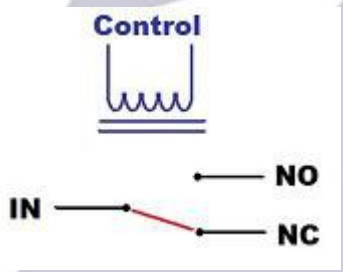
The same applies to simple relays: they can be "normally open" or "normally closed". When a really is unpowered it is in its **normal** state. When it is powered (power is applied to the control side, that is, like a finger push), the state changes. If we name the contacts of the relay IN and OUT, when the relay is closed, current will flow between IN and OUT, because internally IN and OUT will be connected. Conversely, if it's open, no current will flow.

So a **normally open** relay **won't** allow current flow unless powered.

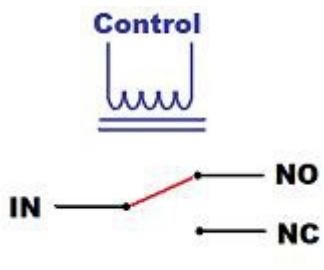
And a **normally closed** relay **will** allow current flow unless powered.

So far so good.

But there are relays a bit more complex (and useful!). They have 3 contacts – let's think of them as IN, NO (for normally open) and NC (for normally closed). The unpowered or relaxed relay will have IN connected to NC - with the ability to change this connection, and short IN with NO when powered (breaking of course the IN to NC connection).



3 contact relay unpowered



3 contact relay powered

For added flexibility, we use both kinds of relays in the Firefly.

2. Planning

Decide exactly what you want to be able to control:

1. Opening / closing the roof
2. Powering the mount
3. powering the CCD Camera(s)

and monitor:

1. position of the mount (at home or not)
2. position of the roof

… remotely. The above list can be considered a bare minimum. Of course it is not the same to setup your backyard observatory so it can be controlled from the living room, than setting up your truly remote observatory 300 Km away from home.

3. Electrical setup

The setup of the Firefly is quite straightforward – it may involve, however, dealing with dangerous voltages and currents (depending on your setup); **please contact a qualified electrician if you don't have the skills needed to perform a 100% safe setup.**

> When naming the state of any relay, we'll think on the "NO" output. This applies to the software, too. So we'll say the relay is OPEN if it's NO output is. All relays will be open when unpowered. Conversely, we'll say the relay is CLOSED if its NO output is connected to the input side.

If you pay a look at the sticker covering the upper side of your Firefly:

… you'll note we've used this naming convention for labeling the input and output connectors.

Second thing to notice: Power Input connectors are at the lower side, Power Output ones at the upper side. Current, be it AC mains (max 240V) or DC from a battery or power supply, will enter the firefly via the Power Input and will be blocked or routed to the Output side. At the Output side we'll have our devices, mount, roof, CCD Camera or whatever.

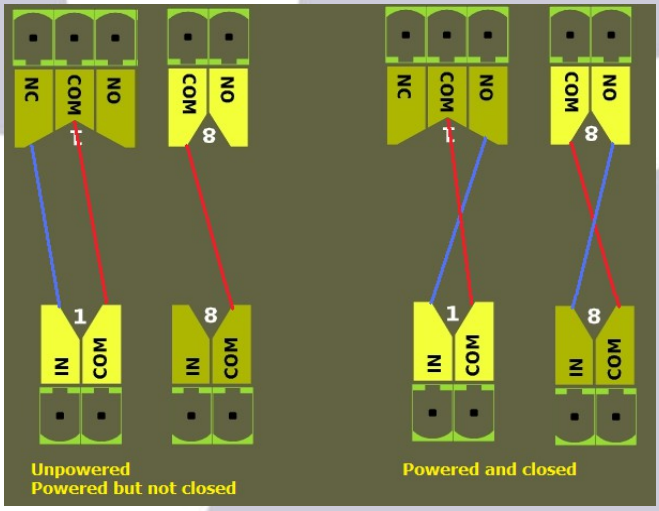
So, to make sure it's clear how the internal wiring goes:

- "COM" will be internally routed from the input to the output side - *always*
- "NO" (Normally Open) will be connected to "In" when the relay is closed by the software
- "NC" (Normally Closed, only available in relays 1 to 4), will be connected to "In" when the relay is open in the software

When the Firefly is unpowered, all the relays will be in the relaxed (normal) state, so:

- All "NC" contacts, present in relays 1 to 4, will be connected to their matching "In" (so they'll be closed)
- All "NO" contacts, will be, of course, just open

Graphically:



So what are the "NC" contacts for? When designing the Firefly, we noticed that certain things should be powered at all times except during the imaging session. One simple example is the observatory lights – you may choose to route the hand switch through the Firefly in such a way that no one can accidentally turn on the lights when you are imaging. You can even use the "NO" contact of the same relay to power on some other thing when the lights go off.

In my case, I have an electrical dehumidifier and I want it off for sure before the roof is opened!

So let's setup things for our minimalist observatory – adding the dehumidifier:

- **Relay 1** → Dehumidifier
- **Relay 2** → Roof control (pulsed – that is, a pushbutton, as standard in many garage door automation motors)
- **Relay 3** → Mount
- **Relay 4** → CCD Camera

Bear in mind! - at the input side, for each relay:

- neutral (AC) or ground (DC) should be wired to "COM" (common)
- phase or live (AC) or positive (DC) should be wired to "In"

So the wiring, input side:

- the same for relays 1, 3, and 4: mains neutral to **COM**, mains phase to **IN**
- for relay 2, as we want to "push a pushbutton", we'll wire **COM** and **IN** together (just a short wire connecting them both). This case is different as we do not need to power the roof motor, but just to push a button.

… and output side:

- Relay 1: **COM** and **NC** go to the dehumidifier (we want it ON by default!)
- Relay 2: **COM** and **NO** go to the rolling roof motor pushbutton
- Relay 3: **COM** and **NO** to the mount power supply.
- Relay 4: **COM** and **NO** to the CCD Camera power supply.

Note: the box of the unit must be connected to ground (earth).

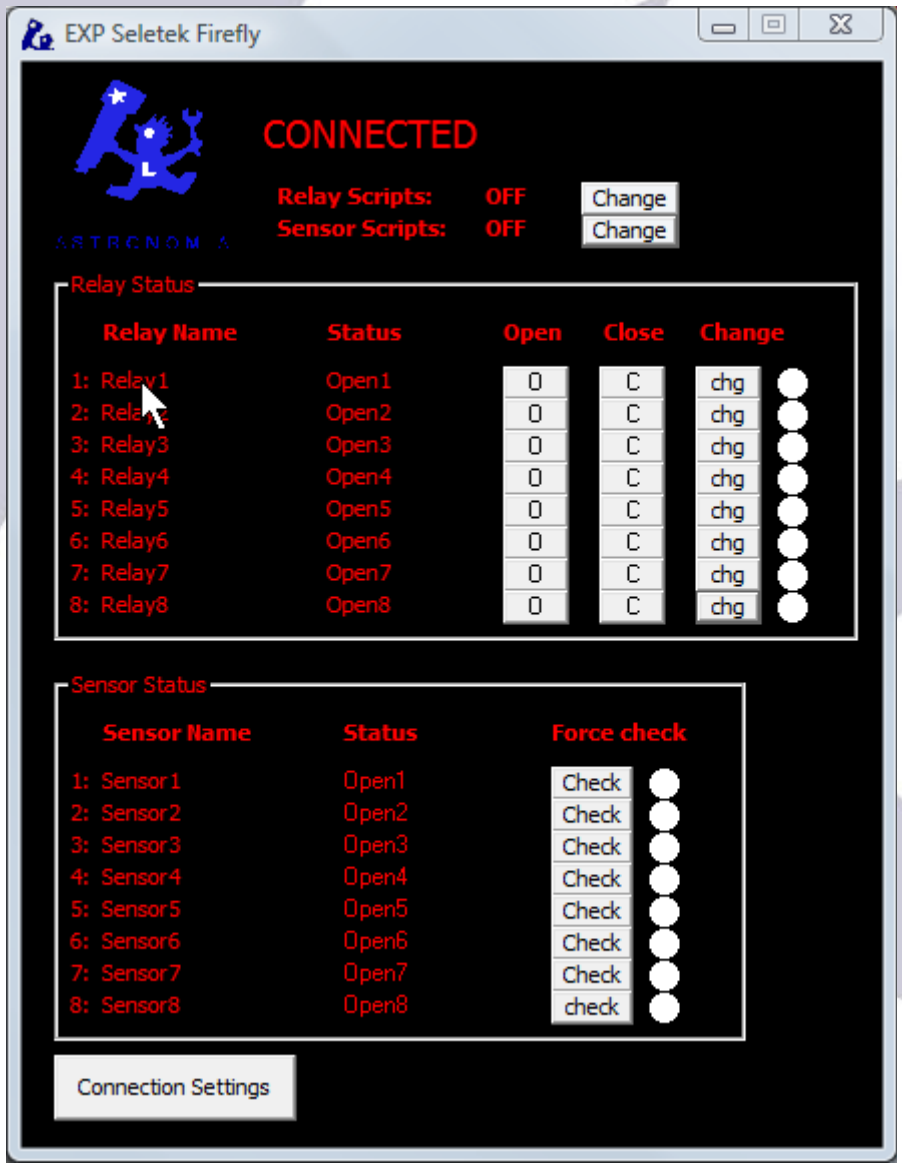
The LED for each relay will be **on** when the relay is **closed**.

4. Software

4-A) Remote control

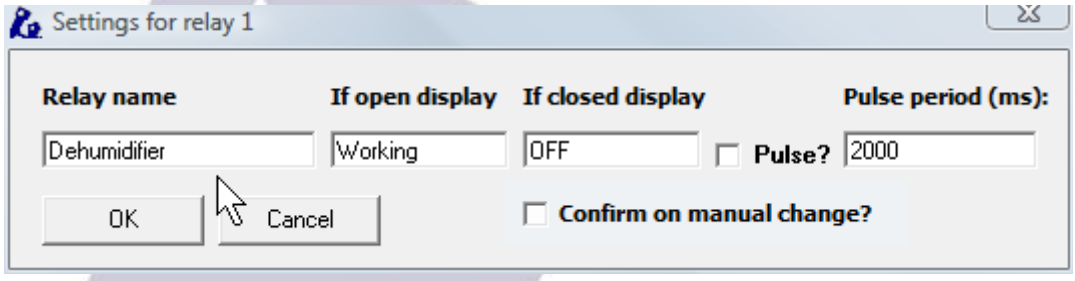
The remote control software supplied with your Firefly allows you to name every physical connection so you don't have to guess. It is really easy to understand and we'll review it working for our minimalist observatory.

Launch the software, and you'll see this window:

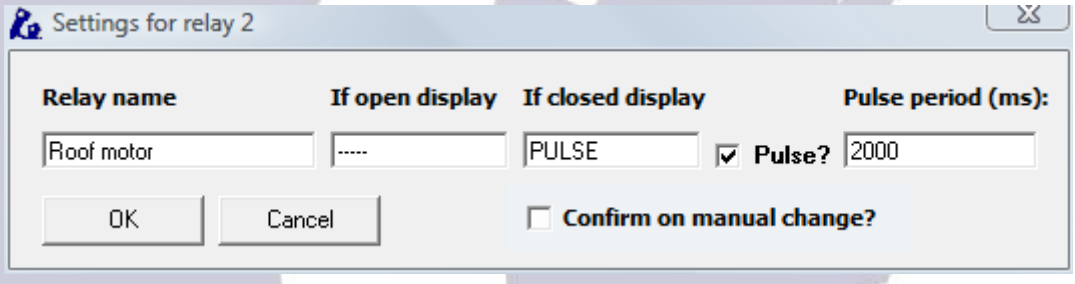


You can give a name to each relay, and even to each state – so relay 4 becomes "CCD Camera", closed becomes "On" and opened becomes "Off".

Just click over the current name (where the mouse cursor is in the above image), and fill:

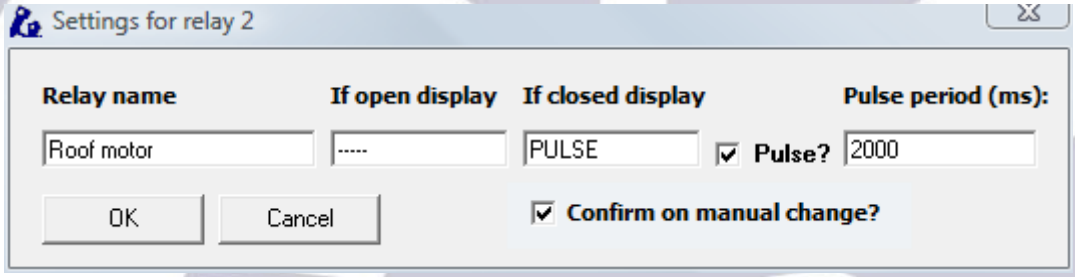


You can name the relay and what should appear when it's closed and open. You can also define it to be a "pulsed" relay, as in the roof motor configuration:



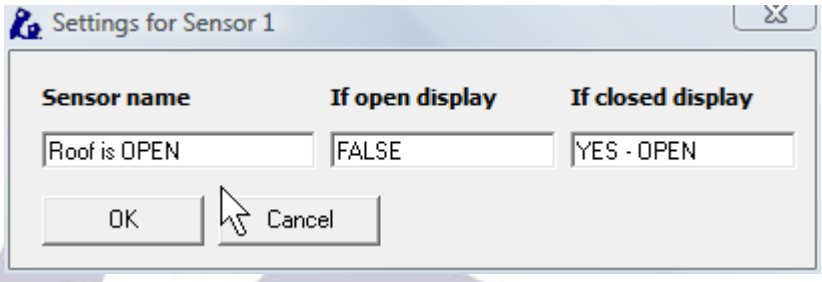
The pulse period, you guessed it, is the approximate duration of the pulse in millisecods.

Last, and to avoid accidental clicks, you can protect any relay selecting...



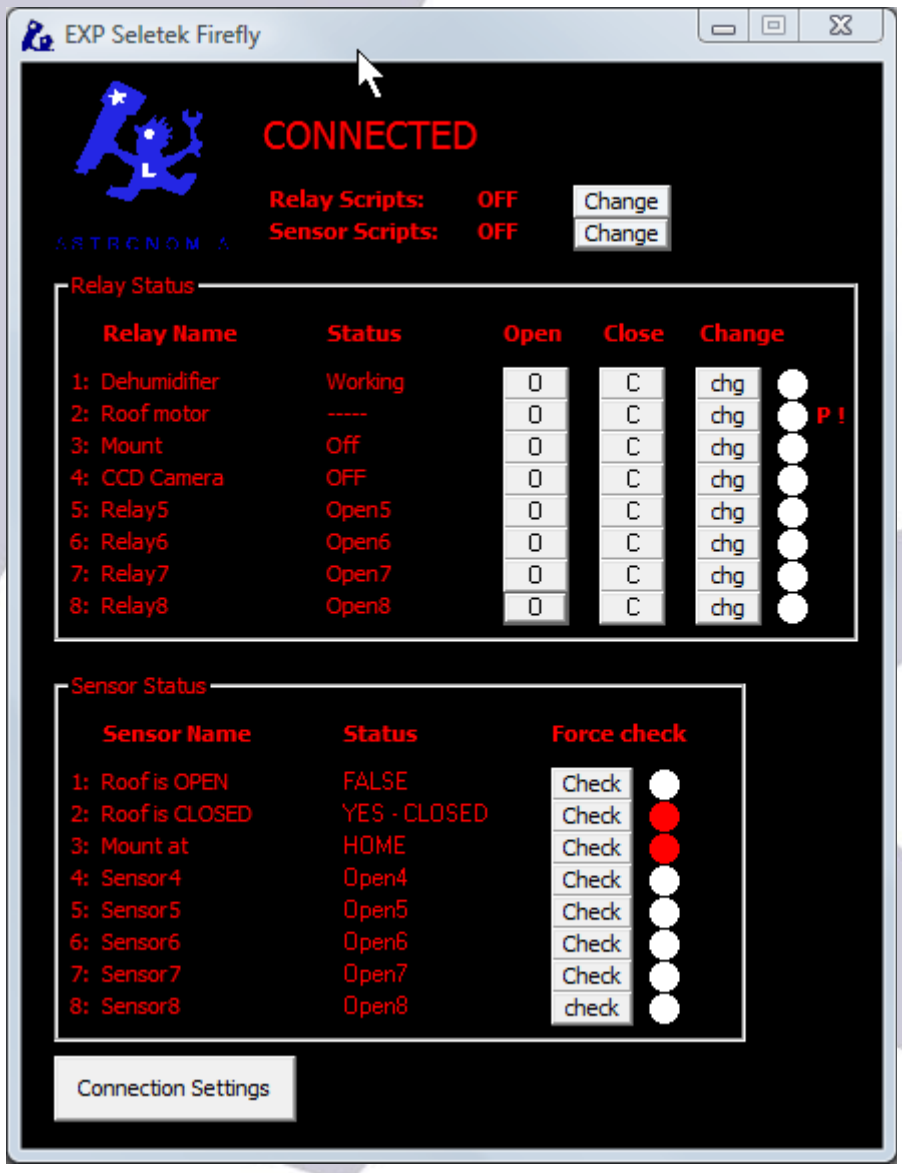
… this way you'll be asked for confirmation if you click any of the buttons concerning that relay.

The same naming applies to sensors:



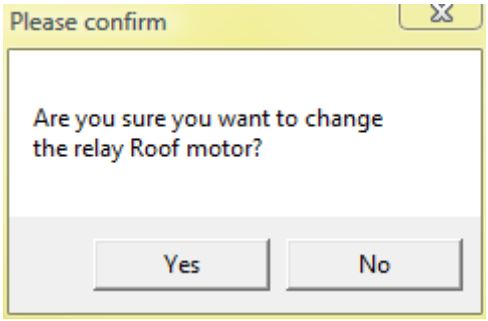
for the roof (roll off roof) we suggest using two switches, one for signaling "open" and other to signal "closed", this way we will notice should the roof stop mid-travel.

So before an imaging session, with our observatory closed our window will look like this:



Dehumidifier working, Roof not opened – but closed, and Mount at HOME position. Great.

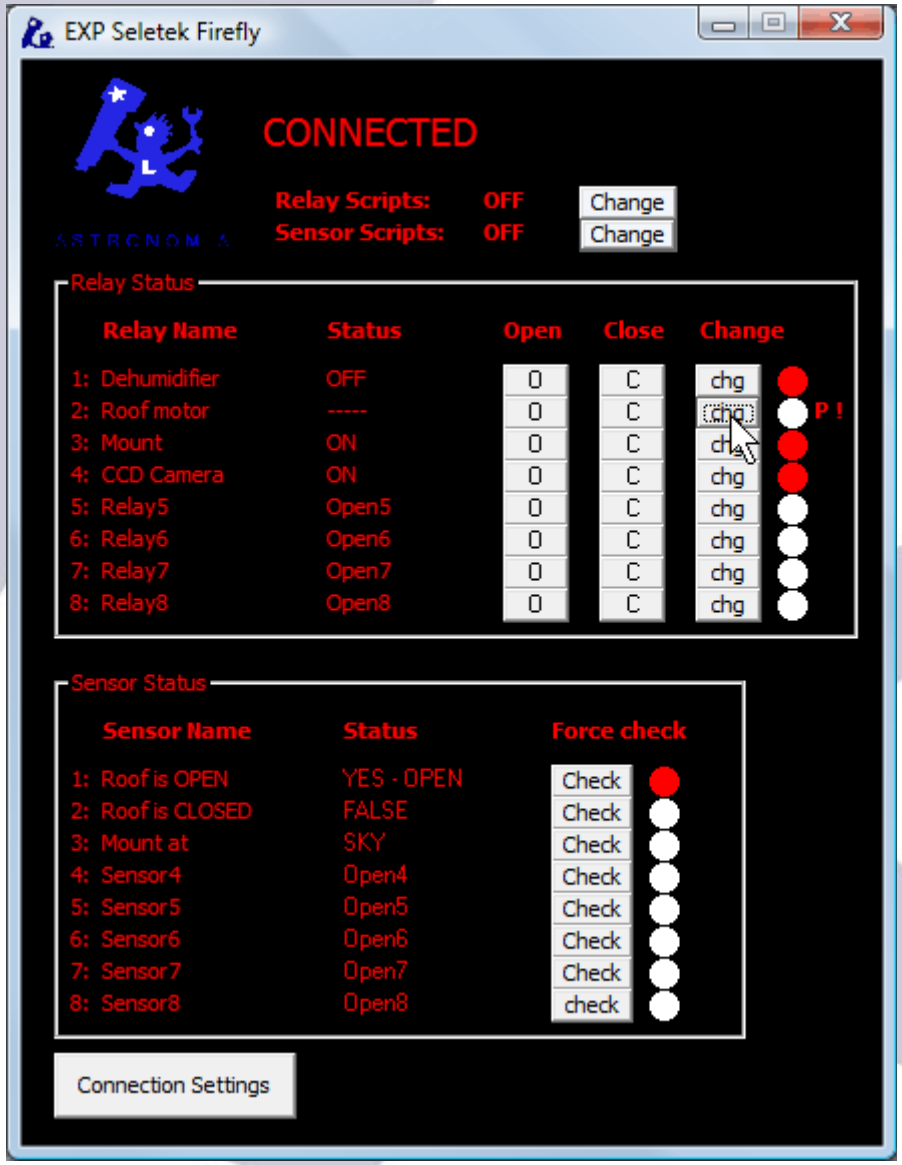
Switch off the humidifier (click "Close" in its line) and click "change" (chg) at the Roof motor line (NOTE: *the pulse we defined earlier will only work when* ***change*** *is clicked; please note the red "P" at the right of the Relay 2*).



As we also selected the "Confirm on manual change" (hence the "!" after the "P"), we'll get a confirmation window:

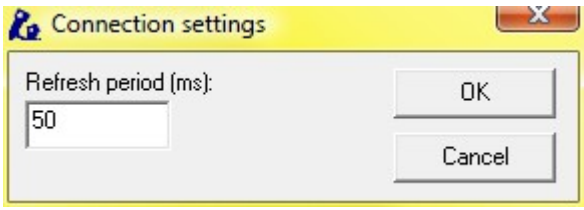
… select yes to proceed. You can now also power the mount and CCD Camera.

The roof will open, and, once fully opened, we'll have this:



… as you can see, it reflects the current state of things.

Last thing worth mentioning is the connection settings:

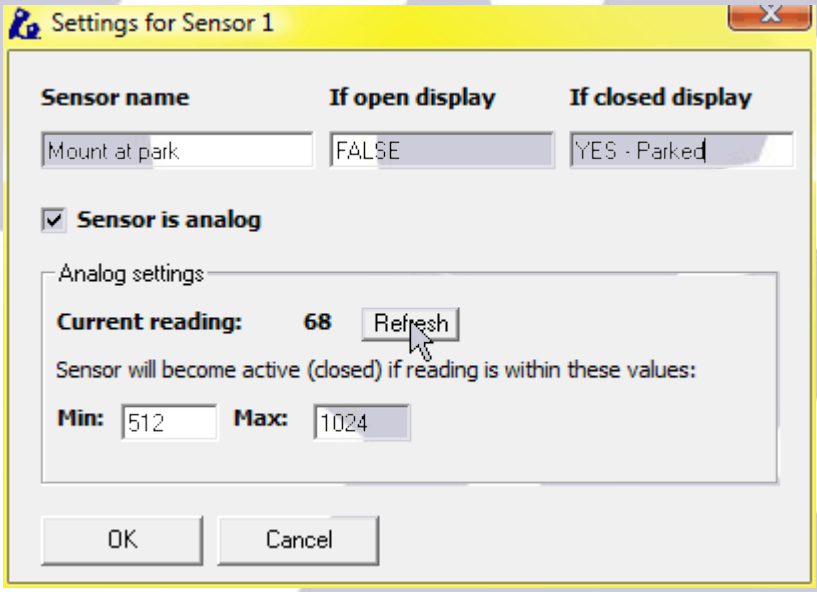


The sensors (and relays) will be checked, one at a time, every 50ms (or any other frequency, between 10 and 60000 ms). You can always force the immediate check of a sensor clicking on the appropriate button "Check".

Addenda: analog sensors

Some sensors, such our new IR distance measuring one, yield analog instead of digital values. This means will get a reading from the sensor, instead of just open or closed, a value between 0 and 1024.

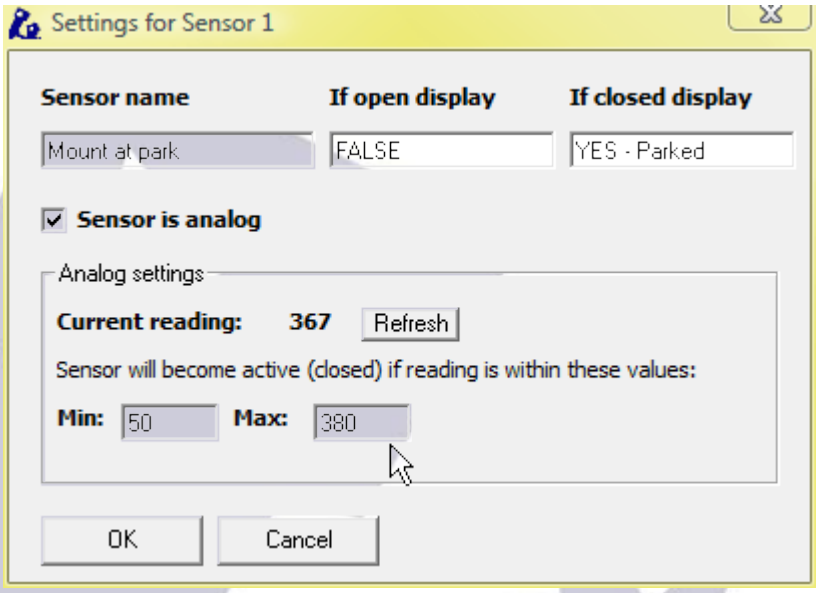
In order to use one of these, you'll have to configure it as analog:



… as the image shows. Clicking "refresh" will update the current reading. The goal is determining the range of values where we can consider the object is in its place; in our example, the mount is parked, so we could:

- unpark the mount: the sensor should read 0 if properly placed, but other values could do, too
- approach the mount to the parking position, clicking refresh and taking note of the values

In a few tries we'll be able to know in what range of values the mount is in a safe position; filling with these values the "Min" and "Max" fields:



… the main Firefly window will update its display accordingly to the current position of the mount.

If you are going to use scripts with an analog sensor, please read the corresponding addenda.

4-B) Automated control - ASCOM

Automation requires a bit more of effort; apart from the Seletek (be it Armadillo or Platypus) / Firefly combo, we'll need some automation software (there are many in the market, such as CCDCommander, CCD Autopilot, and ACP to name the most popular ones), automated focusing support, and, most important, everything running smooth.

The ASCOM standard plays an important role here, as it enables different devices and programs to understand each other. In our case, the Firefly follows the ASCOM standard for domes, with its functionality reflecting that of a roll-off roof observatory.

So basically, thanks to the ASCOM standard, when our automation program wants to close the roof, the Firefly will be asked to do so. Three kind of messages or requests we'll get from the automation program:

- Open the roof
- Close the roof
- Tell me the status of the roof

For each of this messages, the Firefly software will launch a script. All Firefly scripts are located in its home folder (usually "c:\program files\seletek"), under the "ffscripts" folder (that is, "c:\program files\seletek\ffscripts").

Scared? - Don't be!
Scripts can be very simple if the actions they command are simple. There are sample scripts with the Firefly software and we can help if you're stuck!

We have one script for each ASCOM message, so:

- Open the roof: OpenShutter.vbs
- Close the roof: CloseShutter.vbs
- Tell me the status of the roof: ShutterStatus.vbs

Programming the scripts can be a bit scary at first, but is (or can be) simple indeed. It implies, nonetheless, to think in advance and foresee possible situations.

On the plus side, scripts allow for **full customization**, and this is a big plus, worth the effort. For example, we can not only check if the mount is parked before closing or

opening, but also command it to park issuing a few ASCOM calls from the script.
Let's work out the scripts for our sample observatory, using two different ways to program them.

Why there are two approaches? (you can freely skip this explanation)

This question needs to be answered in some detail.

ASCOM calls – when our automation program issues the command to open the roof, for instance – must return very quick. As opening the roof takes time, we cannot just program an OpenShutter script that, say, checks everything is ok, then starts the motor, then waits for it to reach the closed position... this would lock the system.

Typically, any automation program will be asking for the status very often, even each second, and only occasionally asking for the roof to open or close.

A standard conversation between the automation program and the Firefly would look like this one:

Automation program question	**Firefly reply**
Shutter status?	opened
Shutter status?	opened
Shutter status?	opened
…….	……..
Close shutter!	
Shutter status?	Closing
Shutter status?	Closing
Shutter status?	Closing
Shutter status?	Closing
…….	……..
Shutter status?	Closing
Shutter status?	Closed
Shutter status?	Closed
Shutter status?	Closed
Shutter status?	Closed
Shutter status?	Closed
…….	……..

We just said we cannot just make a simple script – with waits, loops, whatever – but this is not the whole truth. Actually *we can*, but let's call it "SyncOpenShutter" instead, and have it called by the ASCOM command in such a way that it will be kept *running in the background*, and the command has the control back immediately.

This is the simple way, and now standard in the Firefly.

We will use the supplied "OpenShutter" script that will just call our, probably customized, script (called SyncOpenShutter, opposite to Asynchronous), and return after calling it. Our SyncOpenShutter will be left running, but **not blocking the system.** So simple.

So for the first approach, the simple one, we just have to program (in most cases just slightly modify) two simple scripts, SyncOpenShutter to open the shutter, and SyncCloseShutter for... yes, closing it.

For the "not so easy" approach: I can't actually foresee any application. It implies actually programming the three scripts (OpenShuter, CloseShutter, and ShutterStatus), in such a way that they will not block the system. This method gives you a sense of control, and may be the ultimate in versatility, but to be honest I'm explaining it as electronic paper is free.

The simple way

This is the most effective way to program the scrips. We just have to program (most probably we'll just use one of examples, and maybe modify it) two scripts, one for opening, other for closing the roof.

The one for opening is to be called "SyncOpenShutter".

The first question is, what do we want to be done before opening the roof? I'd say, check if it is truly closed. If it is, then go on, turn off the dehumidifier and open the roof.

Here's a simple but useful program; *the lines in blue, indented to the right, are not code, just added documentation.*

All lines starting with a single quote ' are considered comments
Write anything after the quote for documentation purposes

```
' Sample FireFly SyncOpenShutter script
' (c) Lunatico Astronomia 2012
```

To avoid errors, any variable we use should be declared first, so...

```
Option Explicit
```

... here we declare the variable Ffly (only one in the program) and set it as a Firefly "Help" object

```
Dim Ffly
set Ffly = CreateObject("FireflyEXP.Help")
```

We start communication with the Firefly, adding a comment before as a reminder

```
' check if closed, closed is sensor 2

if ( not Ffly.SensorDigRead( 2 ) ) then
```

We check if the sensor 2 (roof closed) is active. If not active (not closed),
roof is not closed – and that does not imply it is open!

```
      ' if sensor is OPEN, roof is not closed
      Ffly.AscomShutterStatus = 4   ' ERROR!
```

So if the roof is not closed, we just notify ASCOM there's an error, and nothing more

```
else
```

... in any other case – that is, the roof was closed, we simply go on

```
      Ffly.RelayClose( 1 )    ' dehumidifier off
```

... switching off the dehumidifier

```
      Ffly.RelayChange( 2 )   ' pulse to the roof
      Ffly.AscomShutterStatus = 2   ' opening
```

... and sending a pulse to the roof. We update the ASCOM status to "opening"

```
      wscript.sleep( 20000 )
```

... and wait a reasonable amount of time (20 seconds)
before checking if it has opened.

```
      if ( Ffly.SensorDigRead( 1 ) ) then
            Ffly.AscomShutterStatus = 0
```

```
      else
            Ffly.AscomShutterStatus = 4
      end if
```

Now we check the "open" sensor (1), if it's opened, update ASCOM to "open" (0), else to error (4)

```
end if
```

And that's all!

The code is simple, but maybe there are too many numbers to remember or take into account. It would be easier if we need not remember everything; it's worth the effort to do things like this (additional comments removed now), naming the many numbers so they are easy to remember:

```
' Sample FireFly SyncOpenShutter script
' (c) Lunatico Astronomia 2012

Option Explicit

' ASCOM Constants

const Shutter_Open = 0
const Shutter_Closed = 1
const Shutter_Opening = 2
const Shutter_Closing = 3
const Shutter_Error = 4

' My constants
const Dehumid_Relay = 1
const Roof_Relay = 2
const RoofOpen_Sensor = 1
const RoofClosed_Sensor = 2

const Max_RoofOpenTime = 20000

Dim Ffly
set Ffly = CreateObject("FireflyEXP.Help")

' check if closed

if ( not Ffly.SensorDigRead( RoofClosed_Sensor ) ) then

      ' if sensor is OPEN, roof is not closed

      Ffly.AscomShutterStatus = Shutter_Error

      ' ERROR! - nothing else to be done
else
      ' so it's closed, let's open and do business

      Ffly.RelayClose( Dehumid_Relay )              ' dehumidifier off
      Ffly.RelayChange( Roof_Relay )                ' pulse to the roof
      Ffly.AscomShutterStatus = Shutter_Opening     ' opening
      wscript.sleep( Max_RoofOpenTime )             ' wait for it to open

      ' check again

      if ( Ffly.SensorDigRead( RoofOpen_Sensor ) ) then
```

```
            Ffly.AscomShutterStatus = Shutter_Open
      else
            Ffly.AscomShutterStatus = Shutter_Error
      end if
end if
```

Even if you are not a programmer, modifying this code should be simple. Every Firefly script function will be explained later, but the idea is clear: with the "FireflyEXP.Help" object (Ffly was set to this in the code), we command the Firefly to perform actions, be it check a sensor "Ffly.SensorDigRead" or close a relay "Ffly.RelayClose".

Going back to the examples, the script for closing is very similar, just performing the opposite actions. So, here is SyncCloseShutter.vbs:

```
' Sample FireFly SyncCloseShutter script
' (c) Lunatico Astronomia 2013

Option Explicit

' ASCOM Constants

const Shutter_Open = 0
const Shutter_Closed = 1
const Shutter_Opening = 2
const Shutter_Closing = 3
const Shutter_Error = 4

' My constants
const Dehumid_Relay = 1
const Roof_Relay = 2
const RoofOpen_Sensor = 1
const RoofClosed_Sensor = 2

const Max_RoofCloseTime = 20000

Dim Ffly
set Ffly = CreateObject("FireflyEXP.Help")

' check if opened

if ( not Ffly.SensorDigRead( RoofOpen_Sensor ) ) then

      ' if sensor is OPEN, roof is not opened

      Ffly.AscomShutterStatus = Shutter_Error

      ' ERROR! - nothing else to be done
else
      ' so it's opoen, let's close it and everything else

      Ffly.RelayOpen( Dehumid_Relay )               ' dehumidifier back on
      Ffly.RelayChange( Roof_Relay )                ' pulse to the roof
      Ffly.AscomShutterStatus = Shutter_Closing     ' closing
      wscript.sleep( Max_RoofCloseTime )            ' wait for it to close

      ' check again
```

```
      if ( Ffly.SensorDigRead( RoofClosed_Sensor ) ) then
            Ffly.AscomShutterStatus = Shutter_Closed
      else
            Ffly.AscomShutterStatus = Shutter_Error
      end if
end if
```

And that's all needed to have our simple observatory automated using ASCOM. With this scripts, any ASCOM-aware program will be able to open, close, and report roof status.

Very important

The Firefly software will execute the scripts found at its install folder, "ffscripts" subfolder.
If you program your own SyncOpenShutter.vbs and SyncCloseShutter.vbs scripts, you have to copy them to that folder, ***along with the supplied "OpenShutter.vbs", "CloseShutter.vbs" and "ShutterStatus.vbs" scripts.***

The optional scripts for Relays and Sensor, explained next, also have to be placed there.

The other, more complex way of programming it, is introduced in section 7.

4-C) Automated control – on / off scripts

Additionaly, more scripts can be programmed for the Firefly to launch, adding versatility. As you may have noticed in the control panel:

… you can enable / disable relay and sensor scripts. When enabled, and a change in the state of a relay or sensor is detected, the corresponding script will be launched.

The naming convention is as follows:

RelayClose1.vbs … RelayClose8.vbs: scripts to be executed when the given relay is detected closed.
RelayOpen1.vbs... RelayOpen8.vbs: same, when detected open

SensorOn1.vbs … SensorOn8.vbs: when the sensor is detected ON
SensorOff1.vbs... SensorOff8.vbs: … and detected OFF.

Only existing scripts will be taken into consideration, missing ones will not result in any kind of error.

A simple but powerful application is to have a physical button terminate the session – park the scope, close the roof, switch off the computer. Another possibility is to use one sensor input detect power failures; there's an article explaining this in our web site.

Important considerations:

- The scripts will be executed when (or if) a change is detected. Very fast or spurious changes may not be detected at all (will depend on the refresh period configured).
- They will be executed **just once** for each change.
- Beware of script – script, and script – program interaction, for example:

- If your automation program is in charge of monitoring the weather, do not just close the roof if a sensor detects unsafe weather; but you can set it up so it just closes if after a safeguard period it's still open.

- If you want to switch off the lights (or the dehumidifier) when the roof starts to open, do it at a single place, either at the ASCOM OpenShutter script (will work only when opening via ASCOM) or with a sensor script when the sensor indicating "roof closed" changes to open (will work as long as the Firefly software is running).

Don't forget your scripts can do many more things than handling the Firefly inputs and outputs – they can launch external programs, access any ASCOM objects (such as the mount), command Windows to switch off...

Addenda: analog sensors

Even if you have properly configured the analog sensor settings as explained in the preceding section, that configuration will only affect the remote control panel (main window) of the Firefly.

For the sensors to behave as analog in the scripts, the method "**SensorAnRead**" must be called, and the result (will be between 0 and 1024) matched against the desired values.

5. Firefly functions
(they are properly called methods and properties)

Function	Description	Example(s)
`RelayOpen( relayNumber )`	opens the given relay	`RelayOpen(1)`
`RelayClose( relayNumber )`	closes the given relay	`RelayClose(1)`
`RelayChange( relayNumber )`	changes the relay status, opens it if closed and closes it if open. **Useful because the optional pulse applies to RelayChange**, never to RelayOpen or RelayClose.	`RelayChange( 3 )`
`RelayRead( relayNumber )`	Reads if the given relay is energized / closed (TRUE) or not (FALSE)	
`SensorDigRead( sensorNumber )`	reads the given sensor digitally, that is, returns TRUE or FALSE.	
`SensorAnRead( sensorNumber )`	reads the given sensor analog, that is, returns a number between 0 and 1024.	

`timer( timerNumber )`	Reads or sets a countdown timer in milliseconds. Will stop counting when <= 0.
	It also sets "timerActive(timerNumber) to TRUE" - but will not set it to FALSE when 0 is reached!
		`Timer( 1 ) = 5000`
		`if ( Timer(1) > 0 ) then`
		`' count not finished...`
		`….`
		`end if`

`timerActive( timerNumber )`	Gets or sets the active status of a given timer. Set to TRUE if the timer is started to count	`TimerActive( 1 )`

`flag( flagNumber )`	User flag; can be read or written, to true or false	`Flag( 3 ) = TRUE`
		`if ( flag( 4 ) ) then`
		`…`
		`end if`

`UserVar( varNumber )`	Same as flags, but with integer (numeral) values	`UserVar( 7 ) = 435`
`LogShow`	Shows the log window	
`LogHide`	Hides the log window	
`LogClear`	Erases the log window contents	
`LogAddLine( text )`	Adds the given line to the log window	

`LogAddToLine( moreText )`	Appends the given text to the last written line	
`LogUpdateLine( newtext )`	Changes the contents of the last line of text	
`LogSetBold( TRUE or FALSE)`	Sets or unsets the last line as Bold characters	
`LogSetForeColor( color )`	Sets the color of the text written in the last line	
`LogSetBackColor( color )`	Sets the background color of the text written in the last line	
`AscomShutterStatus( status )`	Sets or reads the status of the shutter – this is accesible to other ASCOM programs.	

Please pay a look at the included sample scripts (explore your software folder, ffscripts, for a "samples" folder) for well documented, clearly written ones.

6. Remote observatory control tips

- Never use fluorescent lights! They are one of the strongest sources of electrical noise.

- Reed (magnetic) proximity switches are very useful to check for roof and mount position. Any will work with the Firefly. A simple one is available from RS components with ref. 289-7783 (don't forget to get the magnet, ref. 289-7812).

- If using pushbuttons to detect roof position (such as RS ref. 746-8605), the usual way is to place the pushbuttons in a fixed position (wall), and the part that will press them – a plastic angle is suggested, as it can bend and won't damage the pushbutton – in the moving roof.

- Plan carefully, be redundant if at all possible, and have an UPS strong enough to close the observatory in case of power failure.

7. The complex method of programming Firefly scripts

If you are reading this section, odds are that you're an advanced programmer and already know what to do. Nonetheless, here's a full intro to asynchronous scripts for the Firefly.

OpenShutter.vbs

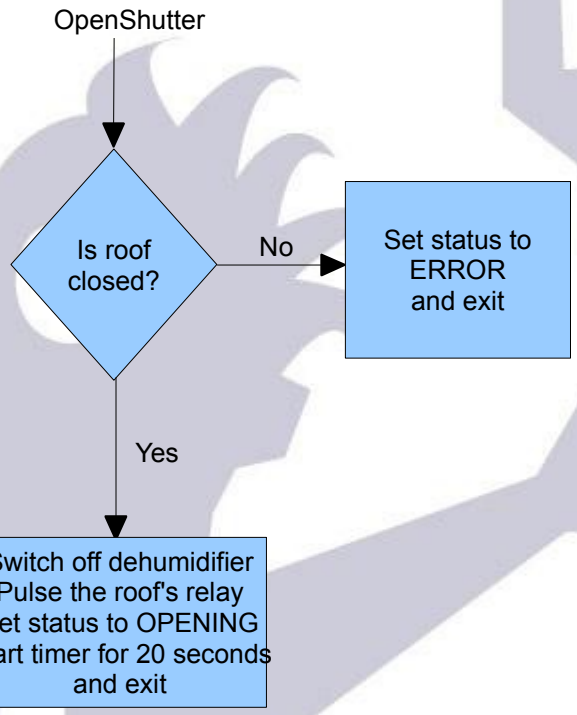
Again, the question is, what do we want to be done before opening the roof? I'd say, check if it is truly closed. If it is, then go on, turn off the dehumidifier and open the roof.

Let's go in deeper detail:

This is the same as before, graphically.

Things worth mentioning:

- whatever happens, we exit
- whatever happens, we update the status.
- We start a 20 seconds timer (let's say the normal period of the roof to open is about 15 seconds, plus a safety margin) to check later.



The actual code for this would be:

```
' Sample FireFly OpenShutter script
' (c) Lunatico Astronomia 2011

Option Explicit


Dim Ffly

set Ffly =
CreateObject("FireflyEXP.Help")
```

All lines starting with a single quote ' are considered comments
Write anything after the quote for documentation purposes

To avoid errors, any variable we use should be declared first, so...

... here we declare the variable Ffly (only one in the program)

```
' check if closed, closed is sensor 2

if ( not Ffly.SensorDigRead( 2 ) ) then
            ' sensors are 1 to 8
      ' if sensor is OPEN, roof is not
closed
      Ffly.AscomShutterStatus = 4   '
ERROR!
else
      Ffly.RelayClose( 1 )    '
dehumidifier off
      Ffly.RelayChange( 2 )   ' pulse
to the roof
      Ffly.AscomShutterStatus = 2   '
opening
      Ffly.timer(1) = 20000   ' timer
1, 20 secs
end if
```

We start communication with the Firefly

Just in case we forget...

We check if the sensor 2 (roof closed) is active. If not active (not closed), roof is not closed – and that does not mean it is open!

As said, if not closed, error.

Ok, it is closed, so let's do the rest of things. Close relay 1,
pulse the relay 2 as programmed in the setup, set status to "opening" (2) and start a timer.

The code is simple, but maybe there are too many numbers to remember or take into account. It would be easier if we need not remember everything; it's worth the effort to do things like this:

```
' Sample FireFly OpenShutter script
' (c) Lunatico Astronomia 2011

Option Explicit

' ASCOM Constants

const Shutter_Open = 0
const Shutter_Closed = 1
const Shutter_Opening = 2
const Shutter_Closing = 3
const Shutter_Error = 4

' My constants
const Dehumid_Relay = 1
const Roof_Relay = 2
const       RoofOpen_Sensor = 1
const       RoofClosed_Sensor = 2
const       Opening_Timer = 1

Dim Ffly

set Ffly = CreateObject("FireflyEXP.Help")

if ( not Ffly.SensorDigRead( RoofClosed_Sensor
) ) then
      ' if sensor is OPEN, roof is not closed
      Ffly.AscomShutterStatus = Shutter_Error
else
      Ffly.RelayClose( Dehumid_Relay )
      Ffly.RelayChange( Roof_Relay )
      Ffly.AscomShutterStatus =
```

The constants relieve us from the burden to remember every number. Type it once, use many. Also help avoid many errors.

Some constants relative to our particular setup

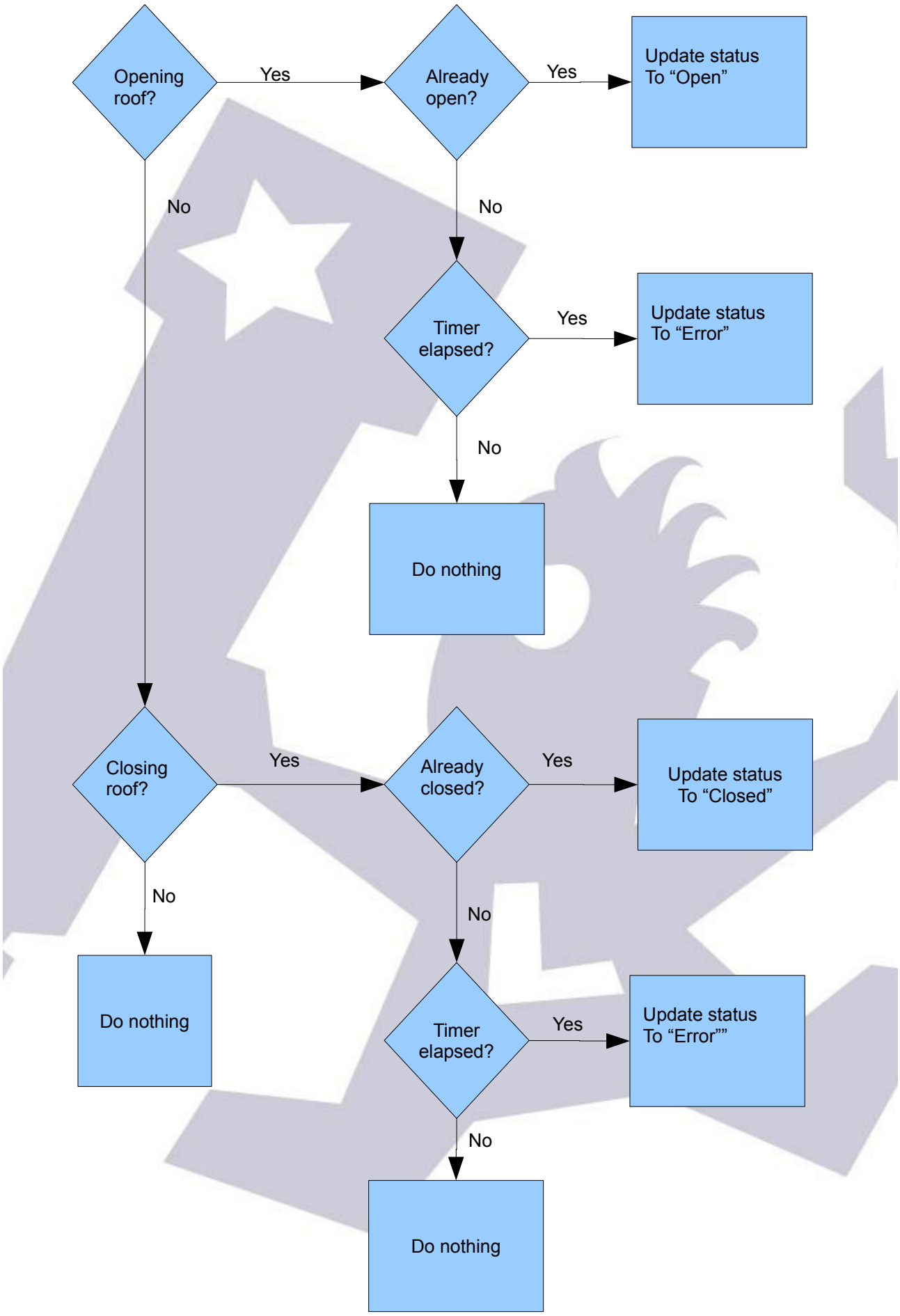
Now the code is so clear not many comments are needed.

```
Shutter_Opening
      Ffly.timer(Opening_Timer) = 20000 ' 20
sec timeout
end if
```

Even if you are not a programmer, modifying this code should be simple within certain limits. Every Firefly script function will be explained later, but the idea is clear: with the "FireflyEXP.Help" object (Ffly was set to this in the code), we command the Firefly to perform actions, be it check a sensor "Ffly.SensorDigRead" or close a relay "Ffly.RelayClose".

The CloseShutter script will be very similar to the OpenShutter one; the different one is the ShutterStatus; it will be called many times by the automation programs to check the current status of the shutter, and we'll use it to confirm our "opened" and "closed" sensors are properly activated.

A typical flowchart of one such script is shown in the next page.


Opening roof?
Yes
Already open?
Yes
Update status To “Open”
No
No
Timer elapsed?
Yes
Update status To “Error”
No
Do nothing
Closing roof?
Yes
Already closed?
Yes
Update status To “Closed”
No
No
Do nothing
Timer elapsed?
Yes
Update status To “Error””
No
Do nothing

… not so complex, but more tricky than the open and close ones.

8. Apendix

Firefly's relays technical specifications:

Nominal switching capacity: 10A @ 125V/250V y 6A @ 277V
Max. switching voltage: 250 V AC, 100 V DC
Max. switching current: 10 A (AC), 5 A (DC)
4 of the relays NC and NO.
Relay model Panasonic JS1-5V-F