**NASA SPACE APPS CHALLENGE**

**Martian’s-Eye**

(A device developed to track the atmosphere on planet Mars.)

**Authors:**

|  |  |
| --- | --- |
| **NAME** | **E-mail** |
|  |  |
| Thanjagari Vamsi | [thanjagarivamsi@gmail.com](mailto:thanjagarivamsi@gmail.com) |
|  |  |
| Prathamesh Dixit | [prathmeshydixit@gmail.com](mailto:prathmeshydixit@gmail.com) |
|  |  |
| Karthik Sanga | [Karthiksanga121@gmail.com](mailto:Karthiksanga121@gmail.com) |
|  |  |
| Udutha Bhargav | [Udutha.bhargav@hotmail.com](mailto:Udutha.bhargav@hotmail.com) |

**Challenge:**

develop sensors to be used by humans on planet Mars.

**Problem:**

It’s only a matter of time before we are able to go to Mars and see, touch and inspect things that have only been visible through the lenses of landers, rovers, and orbiters.The rovers and orbiters currently hovering over Mars are not solidly reliable, moreover there are certain parameters which are quite essential for human existence on Mars which are not properly studied by these devices.We need sophisticated device boarded reliable sensors so that we will have the opportunity to build upon decades of past science experiments from robotic missions, and we will explore familiar and new terrains and environments on Mars in person.

**Abstract:**

It’s only a matter of time before we are able to go to Mars and see, touch and inspect things that have only been visible through the lenses of landers, rovers, and orbiters. The rovers and orbiters currently hovering over Mars are not solidly reliable, moreover there are certain parameters which are quite essential for human existence on Mars which are not properly studied by these devices. We need sophisticated device boarded reliable sensors so that we will have the opportunity to build upon decades of past science experiments from robotic missions, and we will explore familiar and new terrains and environments on Mars in person.

To overcome this problem, our team came up with a subtle solution wherein we use Raspberry Pi and related concepts to solve this issue. We use the Raspberry Pi camera module (v 1.3) which is mounted over the device itself. We also use an object detection sensor and also a temperature sensor which is an added feature which we provide to keep a log of temperature and also hindrances.

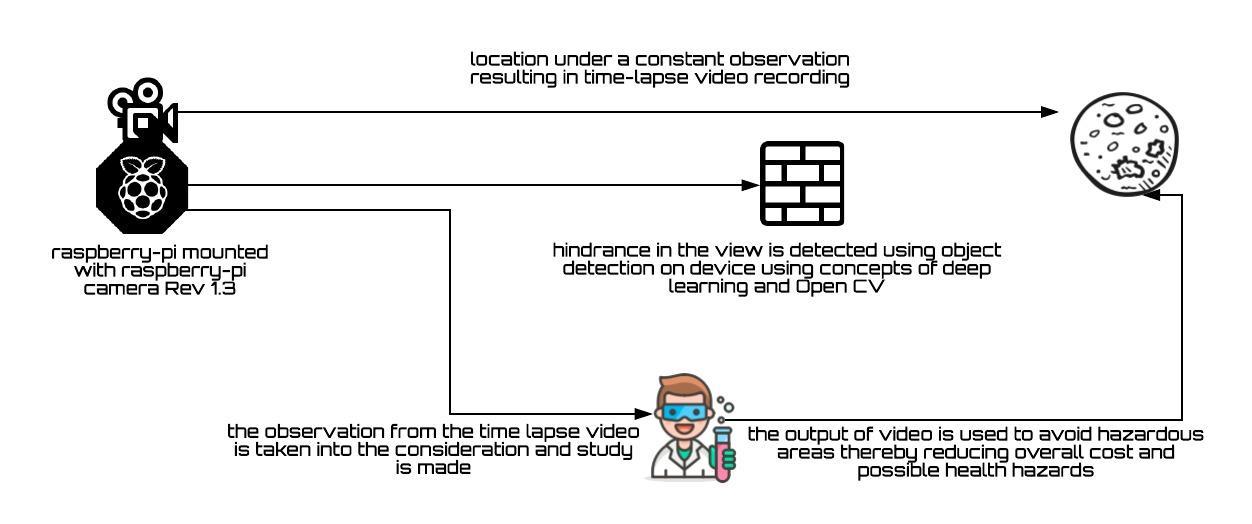
**Technological Stack:**

**Hardware Components:**

* Raspberry Pi
* Raspberry Pi mounted camera module rev1.3
* **SHT20 I2C** temperature & humidity sensor

**Software Components:**

* ImageMagick
* Aurdino
* Python 3 (IDLE)
* Open CV

**UML Diagram and Case Study:** 

**Pseudo code:**

* Open IDLE (menu > programming>python3(IDLE))
* Create a new file (File>New File) and then save it as timelapse.py.
* Now with three simple lines of code, you can use Python to take a photo:
* Save your script, and then run it on your keyboard.
* You can take multiple images using the Camera Module by capturing images with a loop. A for-loop can be used to capture a set number of images.
* Modify your file to incorporate a for-loop. In this example, the Pi Camera will capture 10 images:
* Save your script again and run your program. Then have a look inside your File Manager to see what has been created.
* There’s only one image there, and it’s the last image that was

taken. This is because each image had the same file name, so it was overwritten by the next image to be taken. This is a problem which can be solved by a little modification of the script:

* If you look in the File Manager now, you should see ten images named image0000.jpg up to image0009.jpg.
* The syntax image{0:04d}.jpg adds the value of i, which starts at 0 and ends at 9, to the file name. It also pads the number with zeros, so that there are always 4 digits. This will be important later on.
* Now that you know how to take multiple photos, let’s see how you can turn that sequence into an animated GIF. For this you’re going to need the program ImageMagick. If you haven’t already installed it,

there are instructions in the software setup guide.

* ImageMagick is a command line program that can be used to manipulate images. First of all, you can try it out by opening up your terminal (ctrl+alt+t) and typing the following:
* The -delay option sets the amount of time (in 100ths of a second) between frames. The – loop option sets the number of times the GIF will loop. Here the 0 tells it to loop forever.
* This will take a little time to run, but once it’s complete you should see the file

animation.gif in the File Manager:

**Reducing the file size**

Currently, your animated GIF is probably sitting at around the 10MB range, which is a little large for a ten-frame animation. This is because your Pi camera captures images at a resolution of 3280 x 2464 if you have

* You can double-click this and watch the animation in Image Viewer. Again, give it a little time to open as it’s probably a fairly large file.
* As with all command line programs, you can call ImageMagick from within Python. You just need to use the OS library, as shown below:
* This will take a little time to run. You should see the word done printed in the shell when the script has finished. Your new animation.gif will be playable from the File Manager after a couple of minutes.

a Camera Module v3, or 1920 x 1080 if you have an original Camera Module. You can produce smaller GIFs by using a smaller image resolution.

* Go back to your timelapse.py file. Now add in a single new

line to set the resolution of the images:

* If you want even smaller GIFs,

then choose an even smaller resolution.

The point of time-lapse is to take pictures every few minutes or even hours. To do this you can pause your program between captures, using the time library.

Back in your file, alter the timelapse.py code so that you can import the sleep function, and then pause the script after each capture:

In the above example a picture is taken once every 60 seconds,

**dependencies:**

* Deployment of the device is crucial.
* No interruption of signals should occur.
* Number of preliminary tests should be conducted at the research centre before deploying the device on the planet.
* Sensors should be properly protected and care is to be taken that they are not in any case damaged before deploying.