Impetus: Technical User Manual for a Presence-Detecting Alarm Clock

Sean Webster, A08992738

June 4, 2013

Contents

Chapter 1

Introduction

1.1 Purpose of Impetus

Impetus (literal definition: *the force that makes something happen or happen more quickly*) is an alarm clock that attempts to reduce the possibility of the user to undermine the function of the alarm clock.

An alarm clock's function is to wake the user up in order for them to start their day. In many cases, however, the user can turn off the clock in a way that requires so little effort and attention that they can do so in a subconscious state. The goal of Impetus is to require the user to be active in the deactivation of the alarm while also limiting the complexity of that action.

1.2 Approach

There are roughly three states for Impetus: the idle state, the sleep state, and the alarm state. In each state, the most important feature used is the device's ability to detect human presence. Impetus uses an infrared thermometer sensor to detect the presence of the user. This in addition to scheduling features that will be discussed later make Impetus controllable without physical input.

1.2.1 Presence Detection

For the purposes of this device, detecting a presence is done by comparing the temperature of a distant object with the ambient temperature of the sensor itself. If there is a large enough difference (with the object being hotter), the device assumes someone (or something) is present. How these readings are used vary according to the current state of the device.

Idle State

In the idle state, the task of Impetus is to know at which time to transition to the sleep state. Impetus is naive – it treats any long period of presence detection as good enough to transition.

Sleep State

In the sleep state, Impetus uses temperature readings (as well as other statistics) to generate graphs to present to the user. These metrics are meant to give some insight into sleeping habits, which will be mentioned later. These metrics are sent to the user through the Twitter social messaging service.

Alarm State

As the alarm goes off, if the temperature of the object in front of the sensor is sufficiently above the ambient temperature of the sensor (which is generally room temperature), then the clock will not stop the alarm sequence. Only when the temperature sensed is near the ambient temperature for a set, consecutive amount of time will the alarm state complete. If at any time Impetus detects a presence, the state will reset.

1.2.2 Alarm Scheduling

Scheduling is done through the Twitter. Requests modify the current state of alarms (daily and impromptu alarms), which are saved between sessions as well.

1.3 Justification

1.3.1 Temperature Sensing

Human presence through a contactless IR temperature sensor was chosen as the main method of determining whether or not the alarm should turn off because of what the desired result of an alarm should be – the user should not be anywhere near the alarm clock as it goes off. An alarm has failed if that does not happen. If placed correctly, it should be hard to cheat as well.

Other metrics were considered, like motion detection and range detection, but were found to be missing key pieces of information to make it work for an alarm.

1.3.2 Online Scheduling and Twitter

Online scheduling was chosen to completely get rid of physical interaction with the device (in tandem with presence detection). As something that is used on a daily basis in rarely changing patterns, little interaction should be necessary to use an alarm clock.

Twitter was chosen as both a source for scheduling and a receiver for statistics mainly to simplify the software side of Impetus. Scheduling commands are small (Twitter messages are small), and images are easy to consume on Twitter (graphs of metrics can be saved as images), so while Twitter might not be the most sensible choice for these operations, its nature fits both in some way.

Chapter 2

Hardware

Impetus is comprised of two large components that communicate serially with each other, with small components (sensors, actuators) communicating with one of the large components.

2.1 Large Components

2.1.1 Arduino Pro Mini

The Arduino Pro Mini used in Impetus is responsible for communicating with all the sensors and actuators involved in the device. While most of the communication is done serially, some sensors and actuators have single-pin digital or analog interfaces as well.

The model used in this device has an operating voltage of 3.3V with an 8MHz Atmega328 microcontroller.

2.1.2 Raspberry Pi

The Raspberry Pi mainly communicates state transitions to the Arduino Pro Mini while receiving data from sensors connected to the Arduino to process and send through Twitter.

The model used is a Model B, Revision 2, with 512MB memory and a 700MHz ARM processor. Logic voltage is 3.3V, allowing the Raspberry Pi to communicate with the Arduino Pro Mini without a level shifter.

2.2 Sensors

2.2.1 MLX90614ESF-BCF IR Thermometer

The Melexis MLX90614 family of IR thermometers measure the surface temperature of objects that are a distance away from the sensor. It stores the result in RAM to be accessed through either a PWM or I^2C interface. Impetus communicates with this sensor through I^2C, getting both the object temperature and the ambient temperature for presence detection.

This particular model (BCF) operates at a voltage of 3.3V, and has a field of vision of 10°, which allows the device to get decent measurements from objects as large as the human

head from roughly 5 feet away from the sensor with minor degradation of accuracy due to distance.

2.2.2 ADMP401 MEMS Microphone

This microphone is powered at 3.3V and outputs an analog, oscillating wave from 0V to 3.3V portraying an (amplified) sound wave. In this device, sound is sampled continuously for very short intervals on every state cycle in order find the lowest and highest peaks, which determine volume at that particular interval of time.

2.3 Actuators

2.3.1 7-Segment Serial Display

The 7-segment display is on a breakout board with an Atmega328 which facilitates serial communication with the device through multiple serial interfaces. In Impetus, this actuator communicates over one TTL line (receiving input, never transmitting output) with the Arduino Pro Mini. Simple commands are sent to it to display time and give some form of indication as to which state the device is in. This, along with every other sensor and actuator, operates at 3.3V.

2.3.2 Mini Buzzer

This speaker operates at $3V_{DC}$ with a sound pressure level around 75dB at 20cm. It is toggled on and off by the Arduino Pro Mini during the alarm state, and is off for any ofther state.

Chapter 3

Software[1]

3.1 Arduino Pro Mini

The Arduino Pro Mini's software only deals with communication between sensors and actuators as prescribed by the current state, which it maintains in conjunction with the Raspberry Pi. Its software depends on the Arduino standard library, an I^2C library, and a library that facilitates the creation of serial connections on digital pins.

In addition to the pseudocode for `setup()` and `loop()` below (which in itself ignores helper functions for communicating to sensors), the Arduino Pro Mini code utilizes the Arduino construct `serialEvent()`, which allows incoming serial data to be read as it comes in (after the current `loop()` completes).

3.1.1 Pseudocode

```
setup():
  Setup serial communication with RPi.
  Setup software serial communication with display.
  Setup I2C communication with temperature sensor.
  Read ambient temperature, display on display.

serialEvent():
  while the RPi has more to send:
    Read byte of data.
    If data is the alarm command:
      Change state to the alarm state.
    Else if data is reset command:
      Change state to the idle state.
    Else:
      Write data to the display.
```

[1] All the software written for this device – and the LaTeX source for this technical manual – is maintained at `https://github.com/seandw/impetus-clock` and is licensed under the MIT License.

```
loop():
  If in sleep state:
    Read object temperature.
    Get current volume.
    Compare to values to two stored ones, keep the largest.
    If this has happened 60 times:
      Send stored temperature, volume, ambient temperature to RPi.
      Reset state variables.
  Else:
    If in alarm state and temperature is below ambient threshold:
      Increment count.
    Else if in idle state and temperature is above ambient threshold:
      Increment count.
    Else:
      Reset count.
      Allow buzzer to buzz if in alarm state.
    If in alarm state and count is above 10:
      Stop allowing the buzzer to buzz.
    If count is 60:
      If in alarm state:
        Switch to idle state.
      Else:
        Switch to sleep state.
        Dim display brightness.
        Notify RPi that the current state is sleep.

  If in alarm state:
    Buzz.

  Toggle colon on display, signifying a second has passed.
  Set points on display to signify current state.

  Wait a second.
```

3.2 Raspberry Pi

The software for the Raspberry Pi is divided into three processes that communicate between one another and the Arduino Pro Mini. The parent process is responsible for not only spawning the other two processes, but also to maintain state with the Arduino Pro Mini, receive sensor data, generate graphs, and transmit it to the user via Twitter.

The `updateTime` child process is responsible for maintaining the time on the Arduino Pro Mini. This process shares a serial connection with the Arduino Pro Mini with its parent process, which necessitates a synchronization lock.

The `updateSched` child process is responsible for maintaining the alarm schedule and

other configuration data, and frequently checks the device's Twitter direct messages from the user to update the alarm schedule. This process needs to communicate both configuration data and alarm information to the parent process, which it does through a pipe connection to the parent process.

Outside of the standard Python libraries, this software relies on the `pySerial` library for the serial connection to the Arduino Pro Mini, the `Twython` library for simplifying authentication and calls to Twitter's REST API, and the `matplotlib` library to generate plots from sets of sensor data.

Chapter 4

Operation

Chapter 5

Inventory