# EECS4312 Isolette Assignment

Michael Founk (michaelfounk@outlook.com)
Sean O'Brien (sean1025@my.yorku.ca)

December 10, 2018

**Prism account used for submission**: sean1025

# **Requirements Document**: Temperature control for an Isolette

# Contents



# List of Figures



# List of Tables

# 1. System Overview

The System Under Development (SUD) is a computer controller for the thermostat of an Isolette.[1] An Isolette is an incubator for for an infant that provides controlled temperature, humidity and oxygen (Fig. 1). Isolettes are used extensively in Neonatal Intensive Care Units for the care of premature infants.

This requirements document is specifically for the control of temperature. The purpose of the Isolette computer controller is to maintain the air temperature of an Isolette within a desired range. It senses the current temperature of the Isolette and turns the heat source on and off to warm the air as needed. If the temperature falls too far below or rises too far above the desired temperature range, it activates an alarm to alert the nurse. The system allows the nurse to set the desired temperature range and to set the alarm temperature range outside the desired temperature range of which the alarm should be activated. This requirements documents follows the specification in [1] (Appendix A) except where noted.

Figure 1: Isolette

Many babies have dies due to faulty incubators. There is thus a standard that manufacturers must satisfy. Modern incubators are equipped with alarms for air temperature, skin temperature, oxygen concentration and humidity. The alarms are both visual such

[1] The image in Fig 1 is from: `www.nufer-medical.ch`.

as red warning lamps, and audio such as beep signals. Once measured measured values exceed permitted limits as well as when faults occur in sensors. For one such incident leading to death see "Medical Devices: Use and Safety" shown in Fig. 2.

> **CASE 6:2** **Baby dies through overheating in incubator**
>
> An underdeveloped baby was being treated in an incubator with skin temperature control. When the baby was being washed, the skin sensor was removed and left hanging outside the incubator after the washing. Thus the sensor started measuring the room temperature (approx. 25°C). The control circuits therefore increased the heat to maximum level, and the temperature in the incubator rose to more than 45°C. The baby died.
>
> For increased safety, incubators must be constructed with an extra control circuit that prevents overheating in case the skin sensor is misplaced. The incubator in question was indeed equipped with such a safety circuit, but the circuit was defective.

Figure 2: Incubator Safety Problems [2, p98]

## 2. Goals

The high-level goals (G) of the system are:

- G1—The Infant should be kept at a safe and comfortable temperature.
- G2—The Nurse should be warned if the Infant becomes too hot or too cold.
- G3—The cost of manufacturing the computer controller for the thermostat should be as low as possible.

# 3. Context Diagram

See Fig. A-1 in [1]. The System Under Description (SUD) is a computer *controller* to regulate the temperature of the Isolette. Everything else including the Operator Interface (described in [1]) is in the ecosystem (i.e. in the environment of the controller). The monitored variables and controlled variables for the controller are in Table 1 and Table 2, respectively. For clarity, simplicity and safety, there are some differences between the specifications in this document and the descriptions in [1].[2]

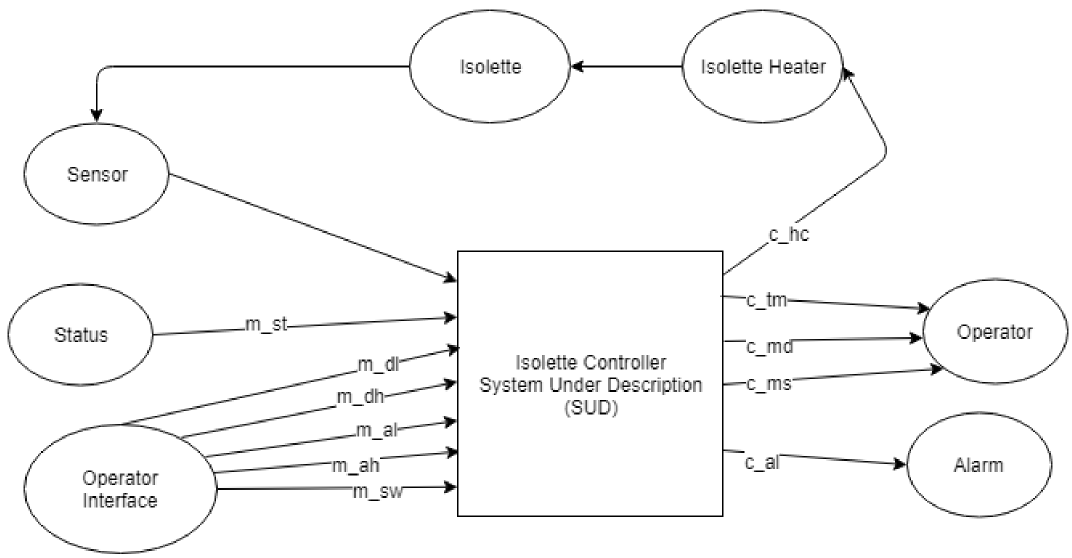



Figure 3: Context Diagram

[2]Documented in the write-up to this assignment: `assign1-spec.pdf`.

# 4. Monitored Variables

The monitored variables are a subset of those described in [1].[3] There is a single status variable *m_st* that is *invalid* whenever any one of the operator inputs or temperature sensor are in a failed state. Otherwise types and ranges are as in [1].

| Name | Type | Range | Units | Physical Interpretation |
|---|---|---|---|---|
| *m_tm* | $\mathbb{R}$ | 68 .. 105 | °F | actual temperature of Isolette air temperature from sensor |
| *m_dl* | $\mathbb{Z}$ | 97 .. 99 | °F | desired lower temperature set by operator |
| *m_dh* | $\mathbb{Z}$ | 98 .. 100 | °F | desired higher temperature set by operator |
| *m_al* | $\mathbb{Z}$ | 93 .. 98 | °F | lower alarm temperature set by operator |
| *m_ah* | $\mathbb{Z}$ | 99 .. 100 | °F | higher alarm temperature set by operator |
| *m_st* | Enumerated | {valid, invalid} | State | status of sensor and operator settings |
| *m_sw* | Enumerated | {on, off} | State | switch set by operator |

Table 1: Monitored Variables

[3] With some change of nomenclature. Monitored variables have an “m” prefix.

# 5. Controlled Variables

The controlled variables are a subset of those described in [1].[4] In addition, there is a mode display *c_md* and a message display *c_ms*.[5]

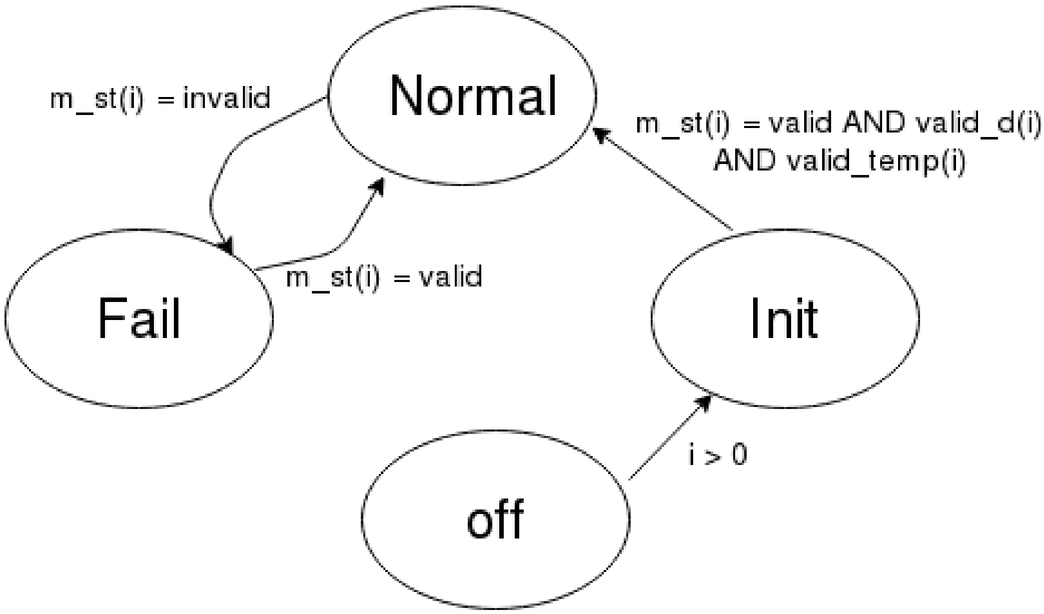
| Name | Type | Range | Units | Physical Interpretation |
|---|---|---|---|---|
| *c_hc* | Enumerated | {on, off} | State | heat control: command to turn heat source on or off |
| *c_td* | $\mathbb{Z}$ | $\{0\} \cup \{68 \,..\, 105\}$ | °F | displayed temperature of Isolette (zero when Isolette is off) |
| *c_al* | Enumerated | {off, on} | State | sound alarm to call nurse |
| *c_md* | Enumerated | {off, init, normal, fail} | State | mode of Isolette operation (failed if $m\_st = invalid$) |
| *c_ms* | Enumerated | {"ok", "er1", "er2", "er3"} | State | messages to display to nurse |

Table 2: Controlled Variables

[4]With some change of nomenclature. Controlled variables have a "c" prefix.

[5]The mode "off" is added to that of Fig. A-4 in [1], and the mode transitions have been changed.

# 6. Mode Diagram




The system is in the off state before it is activated. Once it is turned on it enters the init state. In order for the system to enter the normal state, all three of the conditions must be met at the same time. When the system is in the normal state, and the sensor status is marked as invalid, the system moves into the fail state. If the sensor status is returned to valid, then the system moves from fail to normal. For all states, any permutation of variable values not including the ones specified beside an arrow result in a the state remaining the same. Also the system can enter the Off state from any of the other three states; Init, Normal and Fail.

# 7. R-Descriptions

| | | |
|---|---|---|
| REQ1 | The *controller* shall operate in one of four modes: *off*, *init*, *normal* and *fail*. | See mode diagram on page 6 |

| | | |
|---|---|---|
| REQ2 | In the *normal* mode, the temperature controller shall maintain current temperature inside the Isolette within a set temperature range (the *desired* range). | The *desired* temperature range is $m\_dl$ .. $m\_dh$. If the current temperature $m\_tm$ is outside this range, the controller shall turn the heater on or off via the controlled variable $m\_hc$ to maintain the desired state. |

**Rationale**: The *desired temperature range* will be set by the nurse to the desired range based on the infant's weight and health. The controller shall maintain the current temperature within this range under normal operation.

The following relevant hazard was identified through the safety assessment process:

- **H1**: Prolonged exposure of Infant to unsafe heat or cold;
- *Classification*: catastrophic;
- *Probability*: $< 10^{-9}$ per hour of operation.

To ensure that probability of hazard H1 is $10^{-9}$ per hour of operation, the following derived safety requirement shall apply to the Isolette controller:

| | | |
|---|---|---|
| REQ3 | In *normal* mode, the controller shall activate an alarm whenever<br>• the current temperature falls outside the *alarm* temperature range (either through temperature fluctuation or a change in the alarm range by an operator), or<br>• a failure is signalled in any of the input devices (temperature sensor and operator settings). | The alarm temperature range is $m\_al \,..\, m\_ah$. Monitored variable $m\_st$ shows "invalid" when any of the input signals fail. |

| | | |
|---|---|---|
| REQ4 | Once the alarm is activated, it becomes deactivated in one of two ways:<br>• The nurse turns off the Isolette;<br>• The alarm has lasted for 10 seconds, and after 10 seconds or more the alarm conditions are removed. | Refer to the function table for alarm |

| | | |
|---|---|---|
| REQ5 | In *normal* mode, if a sensor or heater malfunctions, the system transitions to the *fail* mode. | See mode diagram on page 6 |

**Rationale**: If a part of the system malfunctions the system is not operating correctly and must move to the *fail* mode.

| | | |
|---|---|---|
| REQ6 | While the system is in the *fail* mode, the alarm is turned on. | See the function table for the alarm variable |

**Rationale**: The operator must be alerted that the system is not currently functioning.

| | | |
|---|---|---|
| REQ7 | While the system is in the *init* mode, the heater brings the temperature of the incubator into the desired range. | This is required for the system to move from "init" to "normal" |

**Rationale**: Before the infant is put into the Isolette the temperature must meet the conditions supplied by the operator.

# 8. E-descriptions

| | | |
|---|---|---|
| ENV8 | The current temperature received from the sensor is a a real number in the range 68.0 to 105.0°F. | Refer to section 1 |

**Rationale**: This is the specified range of operation of the Isolette. The lower end of this range is useful for monitoring an Isolette that is warming to the Desired Temperature Range. The upper end is set to be greater than the maximum Upper Alarm Temperature.

| | | |
|---|---|---|
| ENV9 | The desired and alarm temperatures received from the operator are all in increments of 1°F. | Refer to the table of monitored variables |

**Rationale**: Marketing studies have shown that customers prefer to set temperatures in 1 degree increments. A resolution 1°F is sufficient to be consistent with the functional and performance requirements specified in the rest of the document.

| | | |
|---|---|---|
| ENV10 | An infant will only be placed in the Isolette in the normal mode | Refer to section 1 |

**Rationale**: The normal mode is the only state that guarantees the conditions that keep the infant safe.

| | | |
|---|---|---|
| ENV11 | The lower alarm temperature will always be less than or equal to the lower desired temperature. | Refer to the monitored/controlled variables tables. |

**Rationale**: If the lower alarm temperature is greater than the lower desired temperature then the alarm can go off while the current temperature is still in the desired temperature range.

| | | |
|---|---|---|
| ENV12 | The lower desired temperature will always be less than or equal to the upper desired temperature. | Refer to the monitored/controlled variables tables. |

**Rationale**: If the lower desired temperature is greater than the upper desired temperature it is unclear if the heat source should be on or off.

# 9. Abstract variables needed for the Function Table

| **Input Conditions** | **valid_temp(i)** |
|---|---|
| m_dl(i) <m_dh(i) AND m_dl(i) <= m_tm(i) AND m_tm(i) <= m_dh(i) | TRUE |
| NOT(m_dl(i) <m_dh(i) AND m_dl(i) <= m_tm(i) AND m_tm(i) <= m_dh(i)) | FALSE |

| **Input Conditions** | **valid_desired(i)** |
|---|---|
| m_al(i) <m_dl(i) AND m_dl(i) <m_dh(i) AND m_dh(i) <m_ah(i) | TRUE |
| NOT(m_al(i) <m_dl(i) AND m_dl(i) <m_dh(i) AND m_dh(i) <m_ah(i)) | FALSE |

# 10. Function Tables

Starting on the next page, provide one function table for each control variable (in Table 2). Each control variable should have its own sub-section heading and its own page.

## 10.1. Function Table for heat control: c_hc

<table>
<tr><th colspan="5">Input Conditions</th><th>c_hc(i)</th></tr>
<tr><td colspan="5">i = 0</td><td>off</td></tr>
<tr><td rowspan="9">i >0</td><td colspan="4">c_md(i-1) = off</td><td>off</td></tr>
<tr><td rowspan="4">c_cm(i-1) = init</td><td colspan="3">m_st(i) = invalid</td><td>off</td></tr>
<tr><td rowspan="3">m_st(i) = valid</td><td rowspan="2">valid_desired(i)</td><td>m_tm(i) >m_dh(i)</td><td>off</td></tr>
<tr><td>m_tm(i) <m_dl(i)</td><td>on</td></tr>
<tr><td colspan="2">NOT(valide_desired(i))</td><td>off</td></tr>
<tr><td rowspan="3">c_md(i-1) = normal</td><td rowspan="2">valid_desired(i)</td><td colspan="2">m_tm(i) >m_dh(i)</td><td>off</td></tr>
<tr><td colspan="2">m_tm(i) <m_dl(i)</td><td>on</td></tr>
<tr><td colspan="3">NOT(valid_desired(i))</td><td>off</td></tr>
<tr><td colspan="4">c_md(i-1) = fail</td><td>off</td></tr>
</table>

## 10.2. Function Table for mode: c_md

| **Input Conditions** | | | | **c_md(i)** |
|---|---|---|---|---|
| i = 0 | | | | off |
| i >0 | m_sw(i-1) = off | | | off |
| | m_sw(i-1) = on | c_md(i-1) = off | | init |
| | | c_md(i-1) = init | m_st(i) = valid AND valid_desired(i) and valid_temp(i) | normal |
| | | | NOT(m_st(i) = valid AND valid_desired(i) and valid_temp(i)) | c_md(i-1) |
| | | c_md(i-1) = normal | m_st(i) = valid | c_md(i-1) |
| | | | m_st(i) = invalid | fail |
| | | c_md(i-1)= fail | m_st(i) = valid | normal |
| | | | m_st(i) = invalid | c_md(i-1) |

## 10.3. Function Table for alarm: c_al

<table>
<tr><th colspan="5">Input Conditions</th><th>c_al</th></tr>
<tr><td colspan="5">i = 0</td><td>off</td></tr>
<tr><td rowspan="7">i >0</td><td colspan="4">c_md(i-1) = off</td><td>off</td></tr>
<tr><td colspan="4">c_md(i-1) = init</td><td>off</td></tr>
<tr><td rowspan="4">c_md(i-1) = normal</td><td rowspan="3">valid_desired(i) AND valid_temp(i)</td><td rowspan="2">c_al(i-1) = on</td><td>held_for(c_al,10)(i-1)=TRUE</td><td>off</td></tr>
<tr><td>held_for(c_al,10)(i-1)=FALSE</td><td>on</td></tr>
<tr><td colspan="2">c_al(i-1) = off</td><td>off</td></tr>
<tr><td colspan="3">NOT(valid_desired(i)) OR NOT (valid_temp(i))</td><td>on</td></tr>
<tr><td colspan="4">c_md(i-1) = fail</td><td>on</td></tr>
</table>

### 10.4. Function Table for temperature display: c_td

| Input Conditions | | c_td |
|---|---|---|
| i = 0 | | 0 |
| i >0 | c_md(i-1) = normal | m_tm(i) |
| | NOT(c_md(i-1) = normal) | 0 |

# 11. Use Case

Related System Goals: G1

Primary Actor: Nurse

Precondition: - The Isolette is in normal mode with an infant inside - The alarm is off

Postcondition:
- The Isolette is in normal mode with new desired and alarm temperature ranges
- The alarm is off
- The infant is inside the Isolette

Main Success Scenario:
- The infant's healthy range for temperature must be adjusted
- The nurse removes the infant from the Isolette
- The nurse turns off the Isolette
- The nurse turns on the Isolette, and specifies the desired ranges
- The Isolette sets the temperature to the desired range
- The nurse places the infant in the Isolette

Alternate Course 1:
- Nurse removes infant from Isolette
- Nurse turns off the Isolette
- Nurse initializes another Isolette
- Nurse places infant in new Isolette when temperature reaches desired range

# References

[1] US FAA. Requirements Engineering Management Handbook. Technical Report DOT/FAA/AR-08/32, U.S. Department of Transportation Federal Aviation Administration, June 2009.

[2] Bertil Jacobson and Alan Murray. *Medical Devices: Use and Safety*. Elsevier, 2007.

# A. Appendix

```
Alarm   [ delta: posreal ]
                : THEORY

  BEGIN

        importing Time[delta]
        importing Mode[delta]
        i: VAR DTIME

        STATE: NONEMPTY_TYPE = {valid, invalid}
    LOW_DESIRE: TYPE+ = subrange(97, 99)
    UPPER_DESIRE: TYPE+ = subrange(98,100)
    LOW_ALARM: TYPE+ = subrange(93, 98)
    UPPER_ALARM: TYPE+ = subrange(99, 103)
        SWITCH: NONEMPTY_TYPE = {on, off}
        TEMP_RANGE: NONEMPTY_TYPE = {r: real | (68.0 <= r AND
        r<= 105.0) OR r = 0}
        MODE: NONEMPTY_TYPE = {off, init, normal, fail}

        m_st : [DTIME -> STATE]
    m_dl : [DTIME -> LOW_DESIRE]
    m_dh : [DTIME -> UPPER_DESIRE]
    m_al : [DTIME -> LOW_ALARM]
    m_ah : [DTIME -> UPPER_ALARM]
        m_sw : [DTIME -> SWITCH]
        m_tm : [DTIME -> TEMP_RANGE]

    c_al: [DTIME -> SWITCH]

        convert_tf(i): bool =
                COND
                c_al(i) = on -> TRUE,
                c_al(i) = off -> FALSE
                ENDCOND

        c_al_ft(i): bool =
                COND
                i = 0 -> c_al(i) = off,
                i > 0 ->
```

```
                              COND
                              c_md(i-1) = off -> c_al(i) = off,
                              c_md(i-1) = init -> c_al(i) = off,
                              c_md(i-1) = normal ->
                                      COND
                                      valid_desired(i) ->
                                              COND
                                              valid_temp(i) -> c_al(i)
                                              = off,
                                              NOT (valid_temp(i)) ->
                                              c_al(i) = on
                                              ENDCOND,
                                      NOT (valid_desired(i)) ->
                                      c_al(i) = on
                                      ENDCOND,
                              c_md(i-1) = fail ->
                                      COND
                                      c_al(i-1) = on ->
                                              COND
                                              held_for(convert_tf,10)(i-1)
                                              = TRUE -> c_al(i) = off,
                                              held_for(convert_tf,10)(i-1)
                                              = FALSE -> c_al(i) = on
                                              ENDCOND,
                                      c_al(i-1) = off -> c_al(i) = on
                                      ENDCOND
                              ENDCOND
                      ENDCOND

  END Alarm
```

```
HeatControl  [ delta: posreal ]
                : THEORY

  BEGIN

        importing Time[delta]
        importing Mode[delta]
        i: VAR DTIME
```

```
        c_hc : [DTIME -> SWITCH]

        c_hc_func(i): bool =
                COND
                i = 0 -> c_hc(i) = off,
                i > 0 ->
                        COND
                        c_md(i-1) = off -> c_hc(i) = off,
                        c_md(i-1) = init ->
                                COND
                                m_st(i) = invalid -> c_hc(i) = off,
                                m_st(i) = valid ->
                                        COND
                                        NOT(valid_desired(i)) ->
                                        c_hc(i) = off
                                        valid_desired(i) ->
                                                COND
                                                m_tm(i) > m_dh(i)
                                                -> c_hc(i) = off,
                                                m_tm(i) < m_dl(i)
                                                -> c_hc(i) = on
                                                ENDCOND
                                        ENDCOND
                        c_md(i-1) = normal ->
                                COND
                                valid_desired(i) ->
                                        COND
                                        m_tm(i) > m_dh(i) ->
                                        c_hc(i) = off,
                                        m_tm(i) < m_dl(i) ->
                                        c_hc(i) = on
                                        ENDCOND
                                NOT(valid_desired(i)) ->
                                c_hc(i) = off
                                ENDCOND
                        c_md(i-1) = fail -> c_hc(i) = off
                ENDCOND
  END HeatControl
```

```
TemperatureDisplay   [ delta: posreal ]
```

```
              : THEORY

BEGIN

      importing Time[delta]
      importing Mode[delta]
      i: VAR DTIME

      c_td : [DTIME -> real]

      c_td_func(i): bool =
              COND
              i = 0 -> 0
              i > 0 ->
                      COND
                      c_md(i-1) = off -> c_td(i) = 0,
                      c_md(i-1) = init -> c_td(i) = 0,
                      c_md(i-1) = normal -> c_td(i) = m_tm(i),
                      c_md(i-1) = fail -> c_td(i) = 0
                      ENDCOND
              ENDCOND

END TemperatureMeter
```