Movescount.com
Powered by Suunto


# SUUNTO APPS

## DEVELOPER MANUAL - MOVESCOUNT.COM

SUUNTO
CONQUER NEW TERRITORY

# TABLE OF CONTENT

# INTRODUCTION

You have probably done your fair share of sports if you are reading this. You have likely spent loads of your time measuring, analyzing and even seeking guidance in the pursuit of hard facts and a deeper understanding of your performance.

And yet you are left wanting. Perhaps you are still missing that one crucial fact that will help you go harder, faster, longer.  We want to change that.

Every one of us has our own unique ambitions. Each sport, each person, each session can have a different set of targets.

We at Suunto have long understood this, and we also understand that sports enthusiasts now their stuff. You know what you really need. Even better, your knowledge could lead to the ultimate tools for like-minded athletes.

With Suunto Apps, we strive to make all that possible. We want you to be able to get the most out of your activities. We want you to innovate, create and change the way sports devices are used. And we want to connect you, with your personal experiences and expertise, to the larger sports community, so that people can also learn to excel.

So, are you ready?

Take the plunge and become a Suunto App developer with the help of this document. We have tried to cover all the key areas you need to know to create the best App you can. If you have any questions, comments or development ideas, please, let us know by sending us your feedback at Movescount.com.

# APP EXAMPLES

## SPRINT COUNTER FOR CYCLING

For cyclists it might be interesting to get a count of how many sprints were done during the workout. This can be calculated with the following App.

First create 2 variables: **Counter = 0** and **Counted = 0.**

```
/* While in sport mode do this once per second */
RESULT=Counter;
if (SUUNTO_BIKE_POWER_AVG[3] > 300 && Counted == 0){
        Counter = Counter + 1;
        RESULT = Counter;
        Counted=1;
}
if (SUUNTO_BIKE_POWER_AVG[3] <= 300 ){
        Counted=0;
}
```

The logical assumption is that each time the cyclist goes over a power level of 300W and maintains that for at least 3 seconds, it considered a sprint. When this has been counted, the same intensive period is not counted again as a new sprint the 3-second average power level drops below 300W. So if you do hard interval sessions, this App will show for example “Peaks 50,” the amount of times when you have been able to keep your power above 300W for at least 3 seconds.

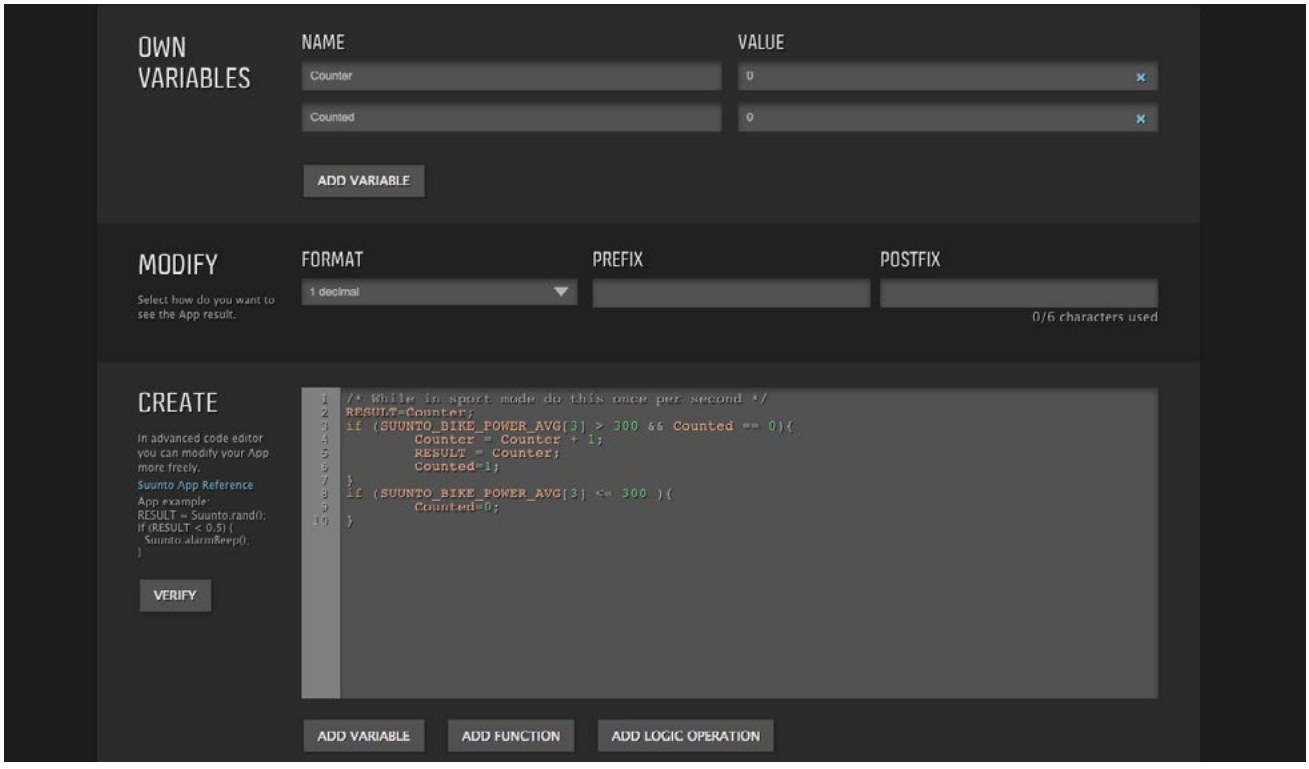


Figure 1 - Example of cycling sprint counting based on watts, buffered value, if-logic, own variables

# COOPER TEST - WITH SMART ESTIMATION

This App shows you the estimation for finish distance until you reach 12 minutes. Estimation is based on your progress, average pace and recent speed. This gives stable estimation, which is reactive to speed changes and time left. After 12 minutes it BEEPS and shows the end result.

Create 6 variables: **raceTime = 720**, **timeLeft = 0, progress = 0, left = 0, Cooper = 0** and **forecastSpeed = 0**

```
timeLeft = raceTime - SUUNTO_DURATION;
progress = SUUNTO_DURATION / raceTime;
left = 1-progress;

forecastSpeed = (SUUNTO_SPEED_AVG[30] * progress) + (SUUNTO_AVG_SPD * left);

if (raceTime == SUUNTO_DURATION){
  Cooper = SUUNTO_DISTANCE*1000;
  Suunto.alarmBeep();
}

if (raceTime > SUUNTO_DURATION){
  RESULT = SUUNTO_DISTANCE * 1000 + ((forecastSpeed/3.6)*timeLeft);
}else {
  RESULT = Cooper;
}
```

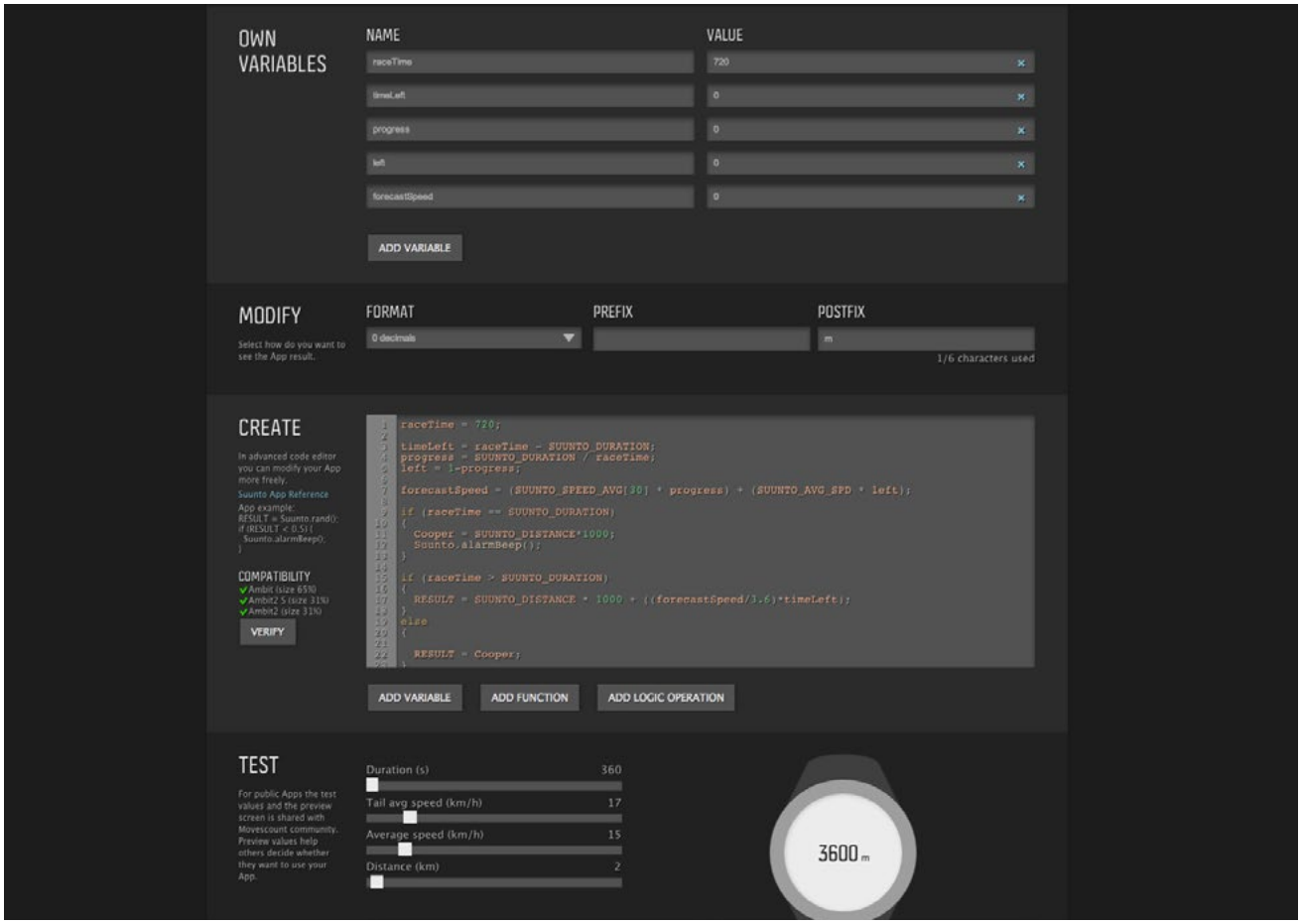


Figure 2 - Estimation App with various own variables

# RACE TIME PREDICTOR 5K - MARATHON

Sometimes you need to show more than one info on screen inside one App. Here's an implementation for that.

First create 3 variables: **print = 0, myCounter = 0** and **estimateScreen = 0.**

```
/* While in sport mode do this once per second*/
/* Make user run 1 km before any estimation is given, show the duration to estimate*/
if((SUUNTO_DISTANCE < 1) && (estimateScreen == 0)){
  print = ((1-SUUNTO_DISTANCE)*1000)/(SUUNTO_AVG_SPD*1000/3600);
  prefix = "Run";
  postfix = "h";
}
if ((SUUNTO_DISTANCE >= 1) && (estimateScreen == 0)){
  estimateScreen = 1;
  myCounter = 0;
}

/*Now when user have ran more than 1km*/
/*Scroll through the different time estimation screens*/
if ((estimateScreen == 1) && (myCounter >= 2)){
  print = SUUNTO_DURATION * Suunto.pow((5000/(SUUNTO_DISTANCE*1000)), 1.06);
  prefix = "5K";
  myCounter = 0;
  estimateScreen = 2;
}
if ((estimateScreen == 2) && (myCounter >=2)){
  print = SUUNTO_DURATION * Suunto.pow((10000/(SUUNTO_DISTANCE*1000)), 1.06);
  prefix = "10K";
  myCounter = 0;
  estimateScreen = 3;
}
if ((estimateScreen == 3) && (myCounter >=2)){
  print = SUUNTO_DURATION * Suunto.pow((21097/(SUUNTO_DISTANCE*1000)), 1.06);
  prefix = "1/2 M";
  myCounter = 0;
  estimateScreen = 4;
}
if ((estimateScreen == 4) && (myCounter >=2)){
  print = SUUNTO_DURATION * Suunto.pow((42195/(SUUNTO_DISTANCE*1000)), 1.06);
  prefix = "Mara";
  myCounter = 0;
  estimateScreen = 1;
}

myCounter = myCounter+1;
RESULT = print;
```

**NOTE:** If you copy the code to code editor, please fix the quotation marks.

# APP BASICS

Each App is described with the following elements.

**App image:** An image is used to showcase the App in App Zone and in your own App library. The image should reflect the nature and content of the App you have created. The text “APP” is automatically overlaid on top of the image.

**App name:** The App name should be descriptive. However, the App name is also used in URL, the shorter the name is, the better.

**Activity:** Choose the main sport for the App. This helps other users find your App.

**Publicity:** To allow your App to be used by others, select “public.” Otherwise, select “private.”

**Category:** Select the category that best fits your App. This category is used to filter search results.

**Description:** The description is very important for others to understand the purpose of your App. Be thorough and use concrete examples whenever possible.

**Tags:** Add tags for your App to help others find it in the App Zone.

**Website:** Use the link field to provide additional information about the App. For example, you can link to research that the App is based on or a website of an event where the App would work perfectly.

**Compatible devices:** Devices where this App can be used are listed here.

**Shoutbox:** Every public App has a shoutbox. This enables potential users to ask questions directly from the developer or from other App users.

**Thumbs up:** Each App can be “liked” by pressing the thumbs up icon. Popularity can be used to sort Apps in App Zone.

**Simulator:** Each App can be simulated in App Designer. The simulator shows what the App result would be in different situations.

**Unique URL:** Each App has a unique URL that can be used on websites, blogs and forums to link to directly to the App page. The address of the app is *www.movescount.com/apps/xxxxID-Appname* where XXXID is the unique ID for the App and APPNAME is the name of the App ad defined by the developer.

**App creator:** App creator shows the name of the person who created the App.

**App create date:** Each App shows the date created.

**TIP!** Share your Apps URL and tell others about your latest App.

# LIFECYCLE OF AN APP

When an App is created, it can have different states within Movescount.com.

**Creation of the App:** Anyone who is registered in Movescount.com can use App Designer to create new Apps. This does not require ownership of an App-compatible device. For example, if you want to create an App for your friend who has such a device, you can do so without owning the device yourself.

**Private Apps:** Only the Movescount.com member who created a private App can utilize it. Private Apps are not viewable at all in App Zone.

**TIP!** When you are creating and testing a new App, it is good to keep it private until you have tested it well enough with your watch. At the moment it's not possible to edit Apps that are used by other members.

**Public Apps:** Public Apps are shown in App Zone. Any registered member can download and use the App. If an App from App Zone is stored in a member's own library, the App cannot be deleted from Movescount.com.

**App in device:** An App stored in a member's library can be used in any compatible device. Compatibility is device specific, so not all the Apps can be used with a given device. Apps are added or removed from the device by customizing sports modes (Gear – Customization –sport modes).

**Logged Apps:** When an App is taken into use, the App results are logged according to the device-specific logging functionality. Logged results are shown as graphs for the recorded Move when uploaded to Movescount.com.

**Apps in library:** The App library is your personal storage area for Apps. All the Apps you have created are automatically stored in your own App library. Public Apps in App Zone can be saved by Movescount members in their own App libraries.

**Removing logged App data from Move:** Logged App data cannot be deleted from a recorded Move without deleting the whole Move from Movescount.com.

**Deleting App from Movescount.com:** An App can be deleted from Movescount.com only if no other member has it in his or her App library, the App is not in use by a device (not in any user's sport mode) and logged results from the App are not stored in any Moves. An App remains in Movescount.com as long as there is at least one person using the App.

**Removing App from App Zone:** When the create of a given App creator deletes the App from his or her own App library, the App is also removed from the App Zone. Other members cannot take the App into use anymore. However, members who have earlier saved the App to their App library will still be able to use it.

**Public App to Private App:** If App was public originally and then changed to private, it will not be shown in App Zone anymore. However, members who have earlier saved the App to their App library will still be able to use it.

# APP DESIGNER

You can design Apps in two different modes.

- **The graphic mode** offers the possibility to easily create mathematical equations with different variables, math operators (+,-,x,/) and structures (parentheses) without any coding.
- The **advanced mode** offers the possibility to do more complex equations with various logical operators and functions.

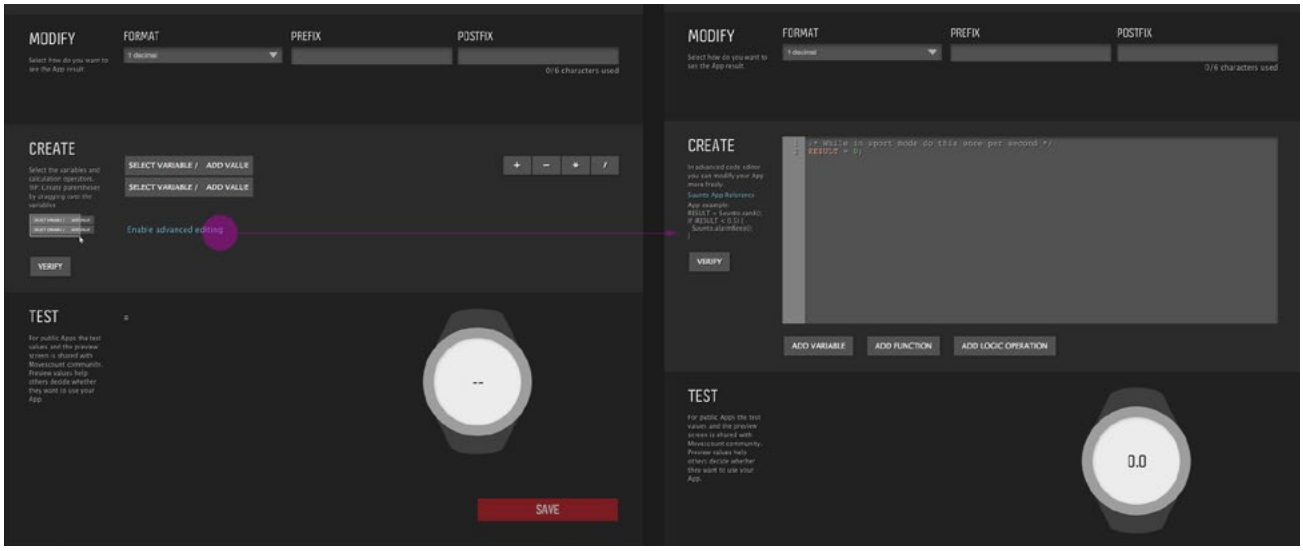


Figure 3 - App Designer: Graphic mode and advanced mode

# APP ZONE

App Zone can be found at [www.movescount.com/Apps](www.movescount.com/Apps). From App Zone you can find all public Apps made by other Movescount.com members.

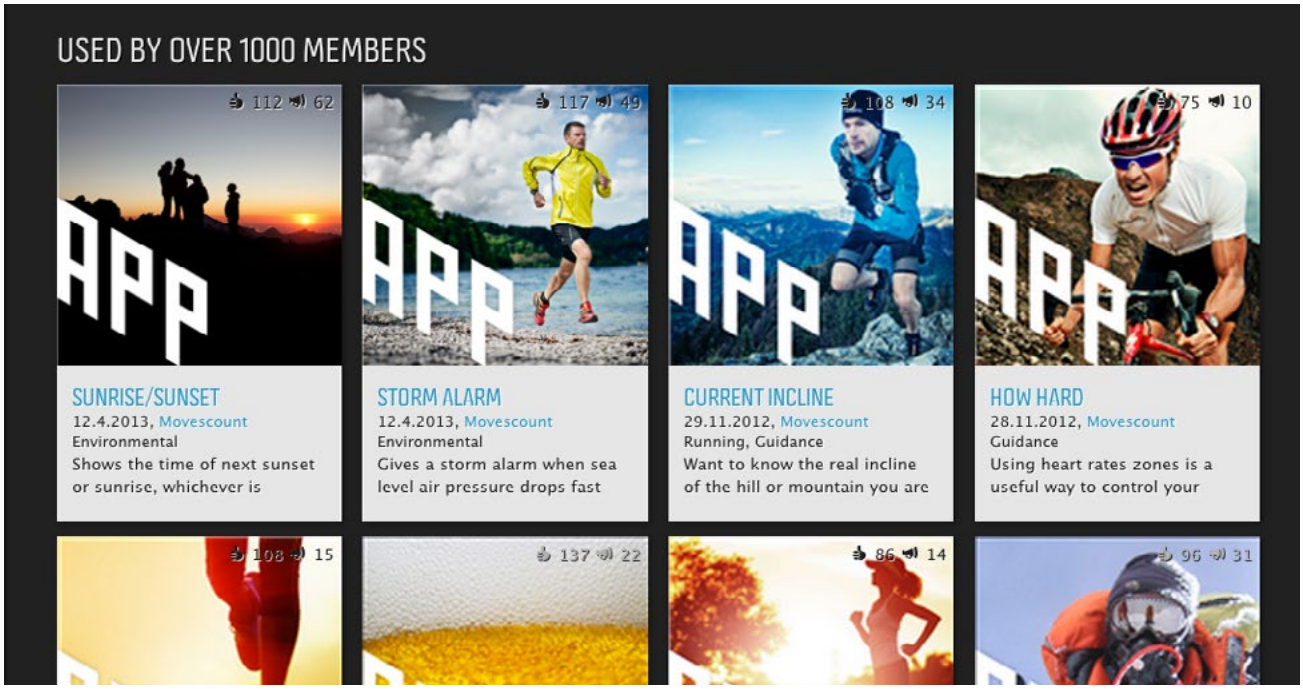


Figure 4 - App Zone in Movescount.com

# USING APP IN DEVICE

Clicking on an App in the App Zone takes you to the home page for that App. There you can find a list of compatible devices for the App. Click the “Save App” button to take the App into use. Any App you save is stored in your personal App library. Once an App is in your library, you can select the App when customizing your devices sports modes.

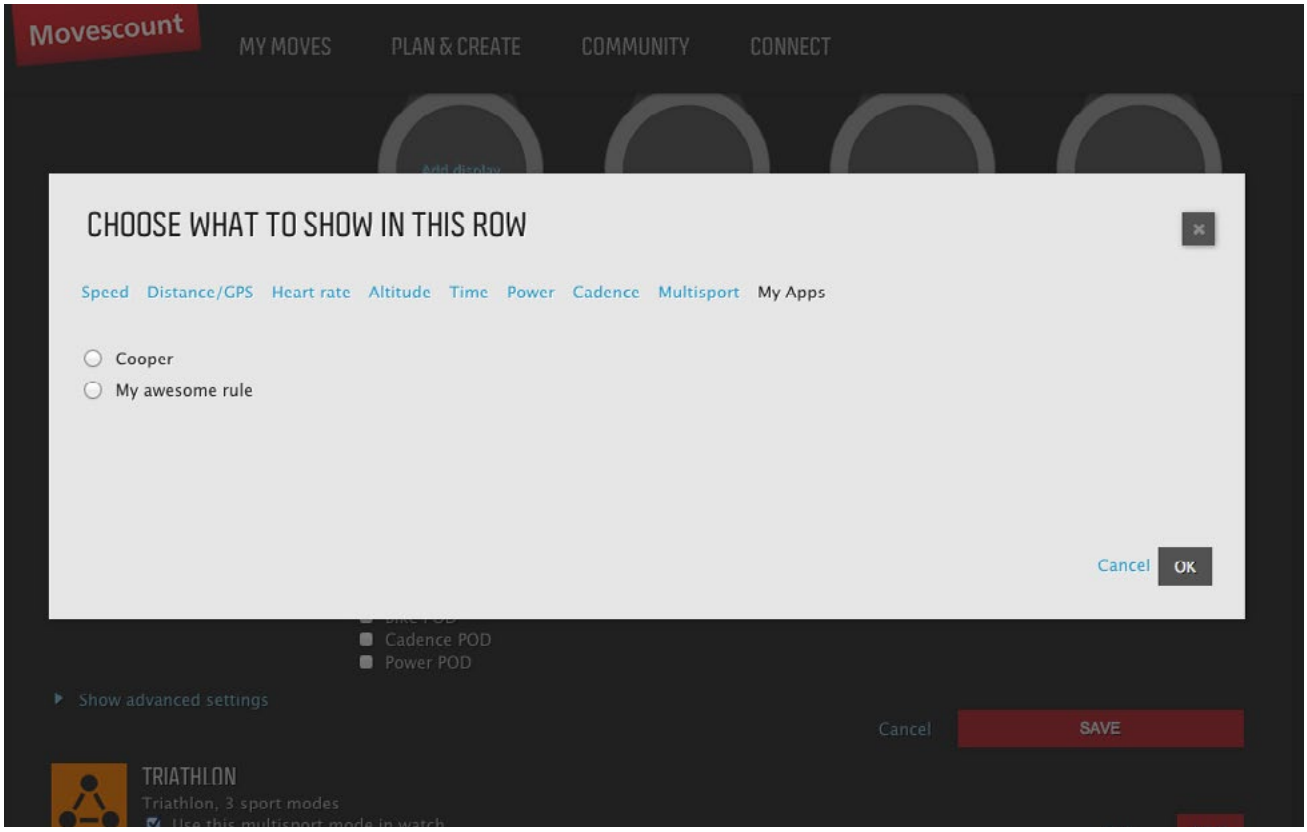


Figure 5 - Selecting App into device in Sport mode customization

# LOGGING APP RESULTS

Apps can be logged as part of a Move recording. Logging App results can be automatic or modified in device settings.

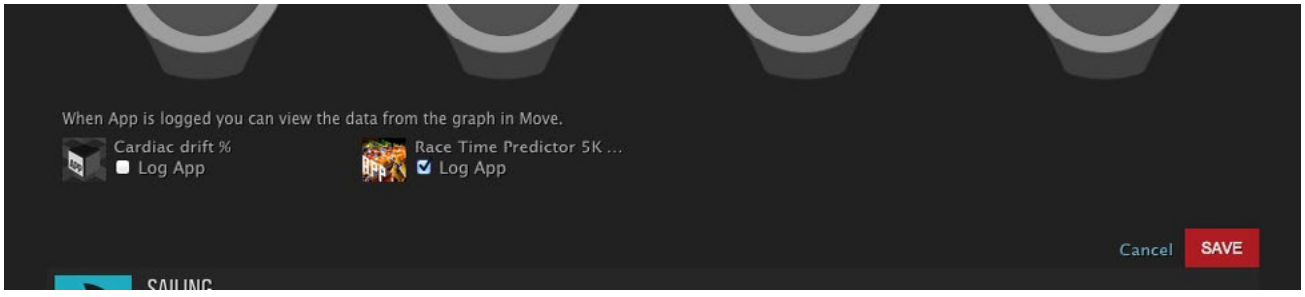


Figure 6 - Logging options for App in Sport mode customization

When the App results are logged, you analyze the data once you have uploaded Move to Movescount.com.

The example below shows the results of an App that calculates running efficiency while running. The App name is shown below the graph as one of the view options. The Y-axis has the prefix “Effic.” and when an individual data point is highlighted, you can see the App details over the graph.

The logged App data is shown in the same way as any other data that Move has. In an App with GPS data record, color coding indicates the App data on the track, and you can select any portion of the graph to see the data distribution as a bar chart as well as a curve.

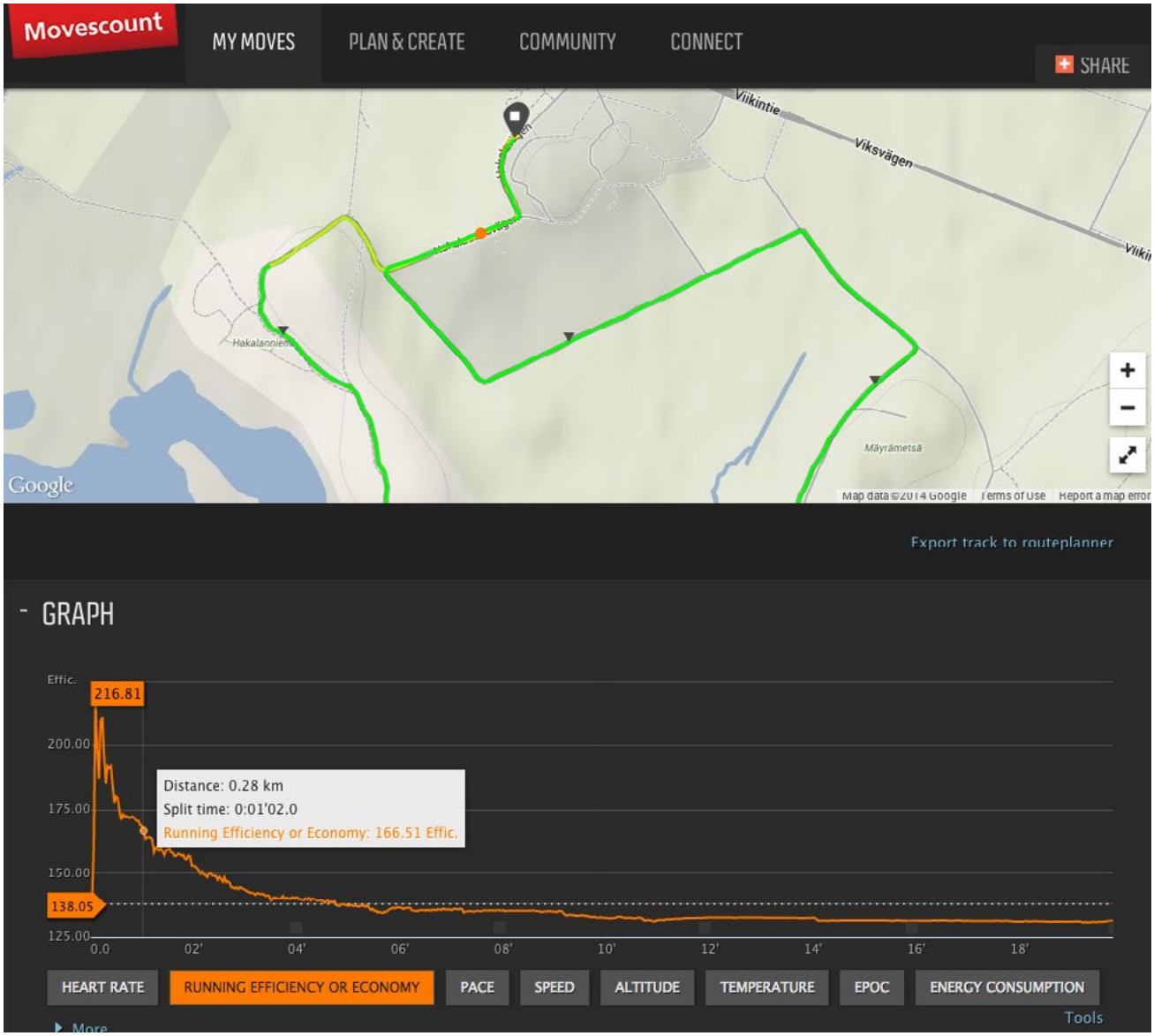

Figure 7 - Running Efficiency or Economy -App result shown in graph and on track.

## APP USER AGREEMENTS

When you create an App and store it in Movescount.com, you are obligated to accept the terms of service.

Likewise, when Movescount.com members save an App from App Zone, they are obligated to accept the end user agreement.

Suunto App Licence Agreement

# APP CODE REFERENCE

## RESULT

The App value is placed in RESULT. RESULT is shown on device screen as a number or graph depending how the App user has customized the sport mode.

For example, the following line would show “100” on the screen:

*RESULT = 100;*

And the following would show current speed on the screen:

*RESULT = SUUNTO_SPEED;*

RESULT can be shown in four different formats, available from the Format drop-down list. Depending on the option selected, the value 100 could be displayed as:

*time: 0:01’40*
*0 decimal: 100*
*1 decimal: 100.0*
*2 decimals: 100.00*

## PREFIX AND POSTFIX

The App results are shown with a prefix and postfix texts. These can be modified in App Designer modify element. It is also possible to modify the prefix and postfix in advanced mode. The syntax for prefix and postfix is as follows:

*prefix=”A”;*
*postfix=”B”;*

The device screen would display “A [RESULT] B”.

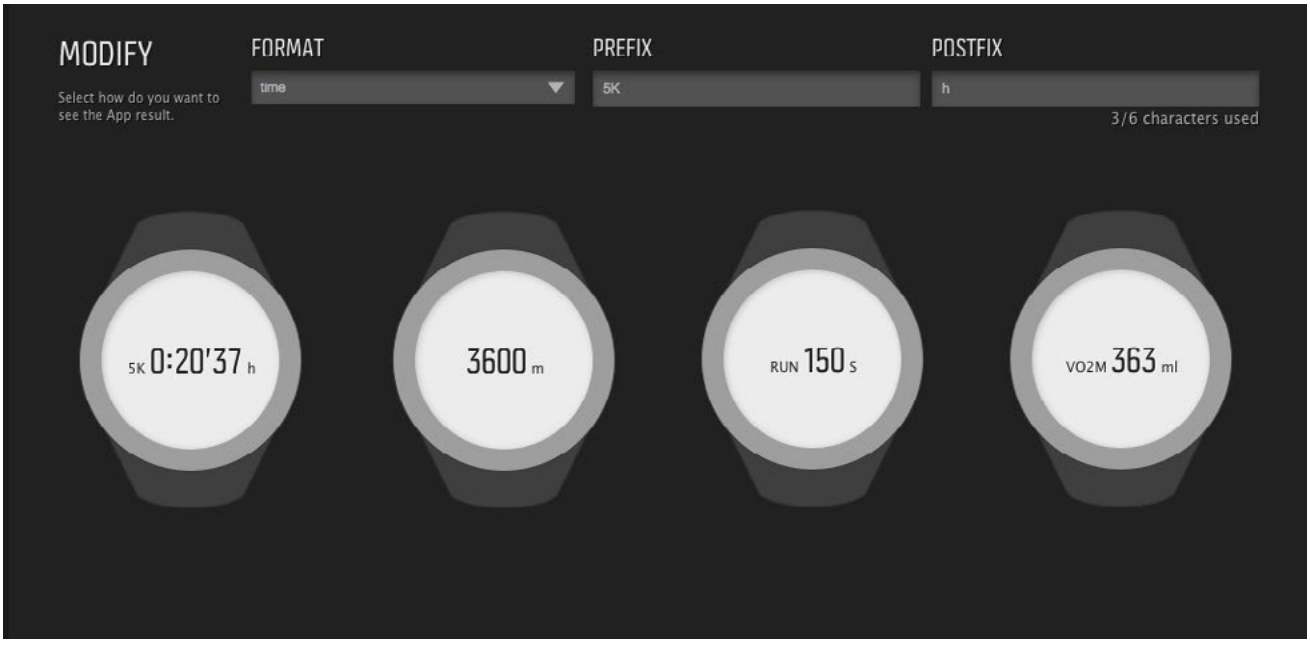


Figure 8 - Result modification with format, prefix and postfix tools.

## CASE SENSITIVITY

App Script is case sensitive.

For instance, the following script shows the result on the screen as “0.”

*RESULT = 0;*

However, this script does NOT work:

*result = 0;*
*Error information: “Whoops! Please check the formula.”*

## SEMICOLON

Every statement needs to be terminated with a semicolon ”;”

The following script shows the result on the screen as “0.”

*RESULT=0;*

However, this script does NOT work:

*RESULT=0*
*Error information: “Compilation has at least one error on line 2.”*

## ERRORS

When creating a new App, you can verify the validity of the script at any stage. The verification will run the code and provide information about errors.

Some of the errors that may be reported include:

**“Compiled binary is too large for any supported device.”**
Error indicates the App is using more memory than what is available in the device. Reduce the number of variables used in the script.

**“The app is used in device which does not support the chosen variables or functions.”**
Error indicates the App has functions/variables that are not supported by the selected device.

**“RESULT must be the output variable.”**
Error indicates RESULT is not defined. Define RESULT in your syntax.

**“Unsupported input variable used.”**
Error indicates App result is not a number. This Error is given if RESULT is a text, or the text is further used as a variable for a function.

**“Unsupported function used.”**
Error indicates the App has a function that is not supported. This error is given if the function name is misspelled, for example.

**“Compilation has at least one error”**
Error indicates that during App compilation there was at least one error. Check the code.

**“Error in buffered variable on line 2”**
Error indicates the variable buffer size is too big. For example SUUNTO_BIKE_POWER_AVG[900]; will create an error becauise the size of the buffer is larger than 30 seconds.

## COMMENTS

Comments must use the following syntax:

```
/* here is a comment */
```

Comments are shown in light gray text.

# MATH FUNCTIONS

Math functions offer various calculation methods to achieve the App result.

## TRIGONOMETRIC FUNCTIONS

### SIN(X)

Sin is equal to the length of a triangle's side opposite the angle divided by length of the hypotenuse.

**Script: retVal = Suunto.sin(argIn)**
• argIn: type float, unit radians
• retVal: type float, no unit, range [-1,1]

### COS(X)

Cos is equal to the length of the triangle side adjacent to angle divided by the length of the hypotenuse

**Script: retVal = Suunto.cos(argIn)**
• argIn: type float, unit radians
• retVal: type float, no unit, range [-1,1]

### ATAN2(Y,X)

Atan2 returns the principal value of the arc tangent of y/x, expressed in radians.

**Script: retVal = Suunto.atan2(argY,argX)**
• argY: type float, no unit
• argX: type float, no unit
• retVal: type float, unit radians, range (-PI,PI]

### TAN(X)

Tan returns the tangent of an angle of x radians

**Script: retVal = Suunto.tan(argIn)**
• argIn: type float, unit radians
• retVal: type float, no unit, range [-INF,INF]

### SIND(X)

Sind(X) returns the sine of the element in X, expressed in degrees.

**Script: retVal = Suunto.sind(argIn)**
• argIn: type float, unit degrees
• retVal: type float, no unit, range [-1,1]

### COSD(X)

Cosd(X) returns the cosine of the element of X, expressed in degrees.

**Script: retVal = Suunto.cosd(argIn)**
• argIn: type float, unit degrees
• retVal: type float, no unit, range [-1,1]

### TAND(X)

Tand(X) returns the tangent of the element of X, expressed in degrees.

**Script: retVal = Suunto.tand(argIn)**

• argIn: type float, unit degrees
• retVal: type float, no unit, range [-INF,INF]

### ATAND2(Y,X)

Atan2 returns the principal value of the arc tangent of y/x, expressed in degrees.

**Script: retVal = Suunto.atand2(argY,argX)**
• argY: type float, no unit
• argX: type float, no unit
• retVal: type float, unit degrees, range (-180,180]

## OTHER MATHEMATICAL FUNCTIONS

### SQRT(X)

Sqrt returns the square root of x.

**Script: retVal = Suunto.sqrt(argX)**
• argX: type float, no unit, range [0,INF]
• retVal: type float, no unit, range [0,INF]

### EXP(X)

Exp returns the exponential value of x.

**Script: retVal = Suunto.exp(argX)**
• argX: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [0,INF]

### LOG(X)

Log returns the natural logarithm of x.

**Script: retVal = Suunto.log(argX)**
• argX: type float, no unit, range [0,INF]
• retVal: type float, no unit, range [-INF,INF]

### LOG10(X)

Log10 returns the common base 10 logarithm of x.

**Script: retVal = Suunto.log10(argX)**
• argX: type float, no unit, range [0,INF]
• retVal: type float, no unit, range [-INF,INF]

### POW(X,Y)

Pow returns x to the power y. If y is not an integer, x must >= 0.

**Script: retVal = Suunto.pow(argX,argY)**
• argX: type float, no unit, range [-INF,INF]
• argY: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [-INF,INF]

### HYPOT(X,Y)

Hypot returns the sqrt(x_x+y_y).

**Script: retVal = Suunto.hypot(argX,argY)**
• argX: type float, no unit, range [-INF,INF]

• argY: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [0,INF]

### RAND()

Rand returns a pseudorandom integer between zero and one.

**Script: retVal = Suunto.rand()**
• retVal: type float, 'unit' integers, range [0,1]

### MOD(X,Y)

Mod returns x - (y _ int(x/y)).

**Script: retVal = Suunto.mod(argX,argY)**
• argX: type float, no unit
• argY: type float, no unit
• retVal: type float, no unit, range (-argY,argY)

### ABS(X)

Abs returns jxj.

**Script: retVal = Suunto.abs(argX)**
• argX: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [0,INF]

### ROUND(X)

Returns the integer value that is nearest to X, with halfway cases rounded away from zero.

**Script: retVal = Suunto.round(argX)**
• argX: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [-INF,INF]

### CEIL(X)

The function rounds X upward, returning the smallest integer value that is not less than X.

**Script: retVal = Suunto.ceil(argX)**
• argX: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [-INF,INF]

### FLOOR(X)

The function rounds X downward, returning the largest integer value that is not greater than X.

**Script: retVal = Suunto.floor(argX)**
• argX: type float, no unit, range [-INF,INF]
• retVal: type float, no unit, range [-INF,INF]

# SUUNTO FUNCTIONS

## ALARMBEEP()

The following syntax plays an alarm.

*Suunto.alarmBeep()*

## LIGHT()

The following syntax turns the backlight on in the device.

*Suunto.light()*

## DISTANCE(LAT, LON)

Returns distance to the current location.

**Script: retVal = Suunto.distance(lat, lon)**
• lat: type float, degrees, range [-90,90]
• lon: type float, degrees, range [-180,180]
• retVal: type float, meters, range 0,21000000]

## DISTANCE(LAT1, LON1, LAT2, LON2)

Returns the distance between two locations.

**Script: retVal = Suunto.distance(lat1, lon1, lat2, lon2)**
• lat1: type float, degrees, range [-90,90]
• lon1: type float, degrees, range [-180,180]
• lat2: type float, degrees, range [-90,90]
• lon2: type float, degrees, range [-180,180]
• retVal: type float, meters, range 0,21000000]

## HEADING(LAT, LON)

Returns the heading from current location to given location.

**Script: retVal = Suunto.heading(lat, lon)**
• lat: type float, degrees, range [-90,90]
• lon: type float, degrees, range [-180,180]
• retVal: type float, degrees, range [0,360]

## HEADING(LAT1, LON1, LAT2, LON2)

Returns the heading from current location to given location.

**Script: retVal = Suunto.heading(lat, lon)**
• lat1: type float, degrees, range [-90,90]
• lon1: type float, degrees, range [-180,180]
• lat2: type float, degrees, range [-90,90]
• lon2: type float, degrees, range [-180,180]
• retVal: type float, degrees, range [0,360]

# OPERATORS AND STRUCTURE

The available operators for an App are:

- `+` *Addition*
- `-` *Subtraction*
- `*` *Multiplication*
- `/` *Division*

**TIP!** When creating mathematical functions in graphic mode, you can switch to the code editor in advanced mode. This will bring the already created function into the code editor automatically.

# RELATIONAL OPERATORS

The following comparisons are available:

- `==` *Equal*
- `!=` *Not equal*
- `>` *Greater than*
- `>=` *Greater than or equal to*
- `<` *Less than*
- `<=` *Less than or equal to*

# LOGICAL OPERATORS

App Designer provides logical operators to join different conditions:

- `||` *OR binary disjunction*
- `&&` *AND conjunction*

Following code is an example how to use logical operators. In this example, a App has different states that are handled with various relational and logical operators.

```
}
/* Executing this if is only allowed if fail state is not enabled */
if (fail != 1) {
  if( SUUNTO_LAP_DURATION >= 40) {
  /* Runner is allowed to run if patrol is not on. This is notified by postfix message "run" */
  if (patrolOn == 0) {
   postfix = "Run";
  }
  /* Distance checker. Runner fails if distance is below zero */
  if((SUUNTO_LAP_DISTANCE * 1000 - SUUNTO_LAP_DURATION * tSpeed * 1000 / 3600) <= 0) {
    fail = 1;
  }
  /* Patrol is initiated between 500 meters. If statement checks if patrol is already on, so that we can avoid executi
  if(SUUNTO_LAP_DISTANCE * 1000 >= (patCount * 500 - 15)) {
    if (SUUNTO_LAP_DISTANCE * 1000 <= (patCount * 500 + 15)) {
    if (patrolOn == 0) {
    if (fail == 0) {
    patrolOn = 1;
    Suunto.alarmBeep();
    postfix = "Patrol";
    patCount = patCount + 1;
    patrolTime = SUUNTO_LAP_DURATION;
    patrolDist = SUUNTO_LAP_DISTANCE * 1000;
```

Figure 9 - Example of relational and logical operators in use

# LOGICAL STRUCTURES

With the advanced mode editor, you can create logical structures such as:

*if (is true){*
*Do this*
*}*

The editor also supports the if…else statements

*if (is true){*
*Do this;*
*}*
*else if (is true){*
*Then do this;*
*}*

In the following example, the App shows different heart rate target levels based on the distance. The first case shows several if statements where the last true statement is shown on the device screen. The target heart rate after the first 3km is 160, so the screen shows 43 beats below the target. The next screen shows the same logic, but with if…else statements. When the first if else statement is true, then the remaining code is not processed and the target heart rate level is set to 130.

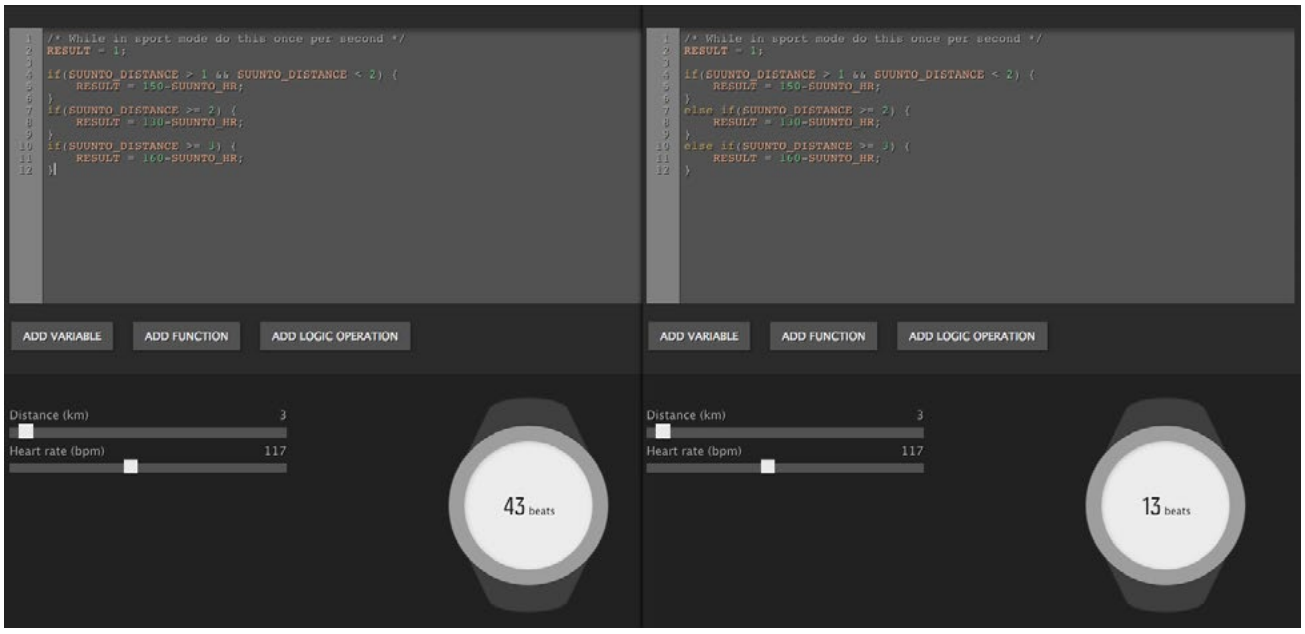


Figure 10 - Several if statements are true and the last if result is shown. Several else if statements are true results first condition to be shown.

# VARIABLES

## OWN VARIABLES

Variables are declared separately. Variables can only store numbers. Each variable that is created needs to have a default value. Variables can be used within the code editor in advanced mode as well as in graphic mode. Variables can be used, for example, as counters when building the App logic.

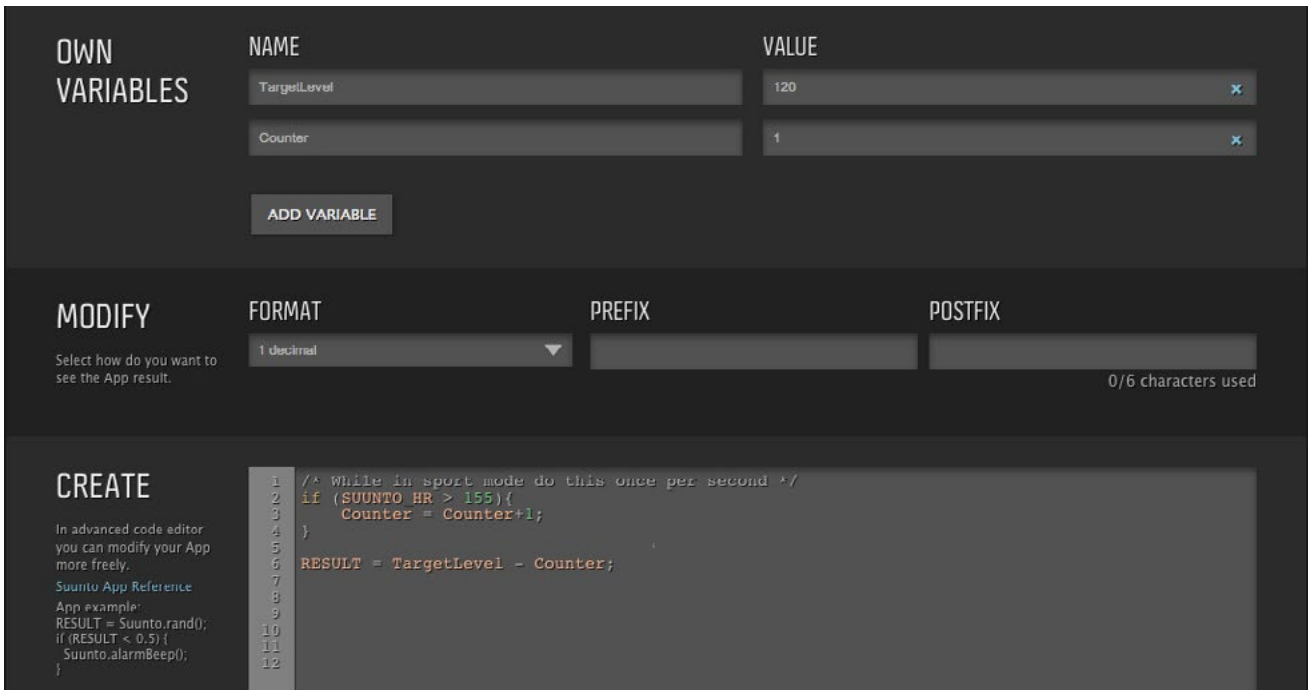


Figure 11 - Own variables used in App Designer and in code editor

## DIFFERENT WATCH VARIABLE TYPES

There are over 200 variables available for designing Apps that follow these basic principles:

### REAL TIME VALUES

These can be current, cumulative or average values, including current average (e.g. average speed at the moment).

### LAP VALUES

Such as average temperature during the lap. Laps may be created manually or automatically (e.g. 1km autolap). Lap values refer to current lap.

### PREVIOUS LAP VALUE

Such as average HR from previous lap. These lap values are taken from the lap that ended when current lap started.

## INSTANT VALUES

Such as instant altitude. Instant value is the value that was recorded when the lap ended. With instant altitude, that would be the altitude when lap button was last pressed.

## TAIL VALUE

Such as average_speed [30]

These are values during the time period “Tail.” This time period is by default 30 seconds, but the duration can be changed shortened. Tail values can be split into five categories:

- Average: average value during the tail
- Max: maximum value during the tail
- Min: lowest or smallest value during the tail
- Diff: difference between first and last value, that is, the value at the first second of the tail compared to the value at the last second of the tail
- Total: total value accumulated during the tail

**NOTE!** Using different tail lengths within one App can lead to a very large App. So, depending to the device, such a large App might not fit into the device memory.

Type of the variables can be:

*s = seconds*

*km/h = kilometers/hour*

*min/km = minutes per kilometer*

*km = kilometers*

*m = meters*

*num = number 0,1,2->*

*bpm = beats per minute (heart rate)*

*PTE = peak training effect (1-5, with one decimal i.e. 2.8)*

*hPa = pressure*

*C = Celsius (temperature)*

*W = Watts (power)*

*rpm = rounds per minute*

*kcal = kilocalories (energy)*

*kg = weight*

*h = hours*

# WATCH VARIABLES

| Variable NAME_IN_EDITOR | Min - Max values in watch (Values used in App designer simulation in parentheses.) | | |
|---|---|---|---|
| **SPEED** | | | |
| Speed SUUNTO_SPEED | 0 | 277 (100) | (km/h) |
| Average speed SUUNTO_AVG_SPD | 0 | 277 (100) | (km/h) |
| Maximum speed SUUNTO_MAX_SPD | 0 | 277 (100) | (km/h) |
| Pace SUUNTO_PACE | 30 (17) | 0.05 (0.6) | (min/km) |
| Lap avg speed SUUNTO_LAP_SPD | 0 | 277 (100) | (km/h) |
| Lap maximum speed SUUNTO_LAP_MAX_SPD | 0 | 277 (100) | (km/h) |
| Lap avg pace SUUNTO_LAP_PACE | 30 (17) | 0.05 (0.6) | (min/km) |
| Previous lap avg speed SUUNTO_LAP_SPD_PREV | 0 | 277 (100) | (km/h) |
| Previous lap max speed SUUNTO_LAP_MAX_SPD_PREV | 0 | 277 (100) | (km/h) |
| Previous lap avg pace SUUNTO_LAP_PACE_PREV | 30 (17) | 0.05 (0.6) | (min/km) |
| Instant speed SUUNTO_SPEED_PREV | 0 | 277 (100) | (km/h) |
| Instant avg speed SUUNTO_AVG_SPD_PREV | 0 | 277 (100) | (km/h) |
| Instant pace SUUNTO_PACE_PREV | 30 (17) | 0.05 (0.6) | (min/km) |
| Tail avg speed[30sec] SUUNTO_SPEED_AVG | 0 | 277 (100) | (km/h) |
| Tail max speed[30sec] SUUNTO_SPEED_MAX | 0 | 277 (100) | (km/h) |
| Tail min speed[30sec] SUUNTO_SPEED_MIN | 0 | 277 (100) | (km/h) |
| Tail diff speed[30sec] SUUNTO_SPEED_DIFF | 0 | 277 (100) | (km/h) |
| Tail avg pace[30sec] SUUNTO_PACE_AVG | 30 (17) | 0.05 (0.6) | (min/km) |
| Tail max pace[30sec] SUUNTO_PACE_MAX | 30 (17) | 0.05 (0.6) | (min/km) |
| Tail min pace[30sec] SUUNTO_PACE_MIN | 30 (17) | 0.05 (0.6) | (min/km) |
| Tail diff pace[30sec] SUUNTO_PACE_DIFF | 30 (17) | 0.05 (0.6) | (min/km) |
| **DISTANCE/GPS** | | | |
| Distance SUUNTO_DISTANCE | 0 | 9999 (50) | (km) |
| Lap distance SUUNTO_LAP_DISTANCE | 0 | 9999 (50) | (km) |
| Autolap distance SUUNTO_AUTOLAP_DISTANCE | 0 | 9999 (50) | (km) |
| Manual lap distance SUUNTO_MANUAL_LAP_DISTANCE | 0 | 9999 (50) | (km) |
| Previous lap distance SUUNTO_LAP_DISTANCE_PREV | 0 | 9999 (50) | (km) |
| Instant distance SUUNTO_DISTANCE_PREV | 0 | 9999 (50) | (km) |

| | | | |
|---|---|---|---|
| Tail total distance[30sec] SUUNTO_DISTANCE_TOT | 0 | 9999 (50) | (km) |
| Latitude SUUNTO_GPS_LATITUDE | -90 | 90 | (degrees) |
| Longitude SUUNTO_GPS_LONGITUDE | -180 | 180 | (degrees) |
| GPS state SUUNTO_GPS_STATE | 0 | 100 | (0=off, 1=starting, 2= hibernating, 3=loading SGEE, 4=activating, 5-100= active) |
| GPS altitude SUUNTO_GPS_ALTITUDE | -10000 (0) | 20000 (5000) | (m) |
| GPS heading SUUNTO_GPS_HEADING | 0 | 360 | (Degrees) |
| Navigation mode SUUNTO_NAVIGATION_MODE | 0 | 2 | (0=off, 1= POI, 2= route) |
| Navigation distance to next waypoint SUUNTO_DISTANCE_TO_NEXT_TARGET | 0 | 41000 (50) | (km) |
| Navigation distance to last waypoint SUUNTO_DISTANCE_TO_PREVIOUS_TARGET | 0 | 41000 (50) | (km) |

## HR

| | | | |
|---|---|---|---|
| Heart rate SUUNTO_HR | 30 | 240 (229) | (bpm) |
| Average heart rate SUUNTO_AVG_HR | 30 | 240 (229) | (bpm) |
| Maximum heart rate SUUNTO_MAX_HR | 30 | 240 (229) | (bpm) |
| Peak Training Effect SUUNTO_PEAKTE | 1 | 5 | (PTE) |
| Energy consumption SUUNTO_ENERGY | 0 | 60000 (3000) | (kcal) |
| Lap avg heart rate SUUNTO_LAP_AVG_HR | 30 | 240 (229) | (bpm) |
| Lap max heart rate SUUNTO_LAP_MAX_HR | 30 | 240 (229) | (bpm) |
| Lap energy consumption SUUNTO_LAP_ENERGY | 0 | 60000 (3000) | (kcal) |
| Previous lap avg heart rate SUUNTO_LAP_AVG_HR_PREV | 30 | 240 (229) | (bpm) |
| Previous lap max heart rate SUUNTO_LAP_MAX_HR_PREV | 30 | 240 (229) | (bpm) |
| Previous lap energy consumption SUUNTO_LAP_ENERGY_PREV | 0 | 60000 (3000) | (kcal) |
| Instant heart rate SUUNTO_HR_PREV | 30 | 240 (229) | (bpm) |
| Instant avg heart rate SUUNTO_AVG_HR_PREV | 30 | 240 (229) | (bpm) |
| Instant Peak Training Effect SUUNTO_PEAKTE_PREV | 1 | 5 | (PTE) |
| Instant energy consumption SUUNTO_ENERGY_PREV | 0 | 60000 (3000) | (kcal) |
| Tail avg heart rate[30sec] SUUNTO_HR_AVG | 30 | 240 (229) | (bpm) |
| Tail max heart rate[30sec] SUUNTO_HR_MAX | 30 | 240 (229) | (bpm) |
| Tail min heart rate[30sec] SUUNTO_HR_MIN | 30 | 240 (229) | (bpm) |
| Tail diff heart rate[30sec] SUUNTO_HR_DIFF | 30 | 240 (229) | (bpm) |

## ALTITUDE

| | | | |
|---|---|---|---|
| Altitude SUUNTO_ALTI | -500 (0) | 9000 (5000) | (m) |
| Ascent SUUNTO_ASCENT | 0 | 65535 (2000) | (m) |
| Descent SUUNTO_DESCENT | 0 | 65535 (2000) | (m) |
| Ascent time SUUNTO_ASCENT_TIME | 0 | 2147483647 (3600) | (s) |
| Descent time SUUNTO_DESCENT_TIME | 0 | 2147483647 (3600) | (s) |
| Vertical speed SUUNTO_VERTICAL_SPD | -212400 (-100) | 212400 (100) | (m/min) |
| Lap ascent SUUNTO_LAP_ASCENT | 0 | 65535 (2000) | (m) |
| Lap descent SUUNTO_LAP_DESCENT | 0 | 65535 (2000) | (m) |
| Lap ascent time SUUNTO_LAP_ASCENT_TIME | 0 | 2147483647 (3600) | (s) |
| Lap descent time SUUNTO_LAP_DESCENT_TIME | 0 | 2147483647 (3600) | (s) |
| Previous lap ascent SUUNTO_LAP_ASCENT_PREV | 0 | 65535 (2000) | (m) |
| Previous lap descent SUUNTO_LAP_DESCENT_PREV | 0 | 65535 (2000) | (m) |
| Previous lap ascent time SUUNTO_LAP_ASCENT_TIME_PREV | 0 | 2147483647 (3600) | (s) |
| Previous lap descent time SUUNTO_LAP_DESCENT_TIME_PREV | 0 | 2147483647 (3600) | (s) |
| Instant altitude SUUNTO_ALTI_PREV | -500 (0) | 9000 (5000) | (m) |
| Instant ascent SUUNTO_ASCENT_PREV | 0 | 65535 (2000) | (m) |
| Instant descent SUUNTO_DESCENT_PREV | 0 | 65535 (2000) | (m) |
| Tail total ascent[30sec] SUUNTO_ASCENT_TOT | 0 | 65535 (2000) | (m) |
| Tail total descent[30sec] SUUNTO_DESCENT_TOT | 0 | 65535 (2000) | (m) |
| Tail total ascent time[30sec] SUUNTO_ASCENT_TIME_TOT | 0 | 2147483647 (3600) | (s) |
| Tail total descent time[30sec] SUUNTO_DESCENT_TIME_TOT | 0 | 2147483647 (3600) | (s) |
| Tail avg altitude[30sec] SUUNTO_ALTI_AVG | -500 (0) | 9000 (5000) | (m) |
| Tail max altitude[30sec] SUUNTO_ALTI_MAX | -500 (0) | 9000 (5000) | (m) |
| Tail min altitude[30sec] SUUNTO_ALTI_MIN | -500 (0) | 9000 (5000) | (m) |
| Tail diff altitude[30sec] SUUNTO_ALTI_DIFF | -500 (0) | 9000 (5000) | (m) |
| Tail avg vertical speed[30sec] SUUNTO_VERTICAL_SPD_AVG | -212400 (-100) | 212400 (100) | (m/min) |
| Tail max vertical speed[30sec] SUUNTO_VERTICAL_SPD_MAX | -212400 (-100) | 212400 (100) | (m/min) |
| Tail min vertical speed[30sec] SUUNTO_VERTICAL_SPD_MIN | -212400 (-100) | 212400 (100) | (m/min) |
| Tail diff vertical speed[30sec] SUUNTO_VERTICAL_SPD_DIFF | -212400 (-100) | 212400 (100) | (m/min) |

## ENVIRONMENT

| | | | |
|---|---|---|---|
| Pressure SUUNTO_PRESSURE | 950 (850) | 1060 (1050) | (hPa - sealevel) |
| Temperature SUUNTO_TEMP | -20 (-30) | 60 (40) | (C) |

| | | | |
|---|---|---|---|
| Minimum temperature SUUNTO_MIN_TEMP | -20 (-30) | 60 (40) | (C) |
| Maximum temperature SUUNTO_MAX_TEMP | -20 (-30) | 60 (40) | (C) |
| Instant pressure SUUNTO_PRESSURE_PREV | 950 (850) | 1060 (1050) | (hPa - sealevel) |
| Instant temperature SUUNTO_TEMP_PREV | -20 (-30) | 60 (40) | (C) |
| Tail avg pressure[30sec] SUUNTO_PRESSURE_AVG | 950 (850) | 1060 (1050) | (hPa - sealevel) |
| Tail max pressure[30sec] SUUNTO_PRESSURE_MAX | 950 (850) | 1060 (1050) | (hPa - sealevel) |
| Tail min pressure[30sec] SUUNTO_PRESSURE_MIN | 950 (850) | 1060 (1050) | (hPa - sealevel) |
| Tail diff pressure[30sec] SUUNTO_PRESSURE_DIFF | 950 (850) | 1060 (1050) | (hPa - sealevel) |
| Tail avg temperature[30sec] SUUNTO_TEMP_AVG | -20 (-30) | 60 (40) | (C) |
| Tail max temperature[30sec] SUUNTO_TEMP_MAX | -20 (-30) | 60 (40) | (C) |
| Tail min temperature[30sec] SUUNTO_TEMP_MIN | -20 (-30) | 60 (40) | (C) |
| Tail diff temperature[30sec] SUUNTO_TEMP_DIFF | -20 (-30) | 60 (40) | (C) |

## TIME

| | | | |
|---|---|---|---|
| Duration SUUNTO_DURATION | 0 | 921600 (18000) | (seconds) |
| Time SUUNTO_TIME | 0 | 86400 | (sec from midnight) |
| Date SUUNTO_DAYS_AFTER_1_1_2000 | 4750 | 6000 | (days since 1.1.2000) |
| Lap number SUUNTO_LAP_NUMBER | 0 | 65535 (50) | (number) |
| Lap duration SUUNTO_LAP_DURATION | 0 | 921600 (18000) | (seconds) |
| Autolap duration SUUNTO_AUTOLAP_DURATION | 0 | 921600 (18000) | (seconds) |
| Manual lap duration SUUNTO_MANUAL_LAP_DURATION | 0 | 921600 (18000) | (seconds) |
| Previous lap number SUUNTO_LAP_NUMBER_PREV | 0 | 65535 (50) | (number) |
| Previous lap duration SUUNTO_LAP_DURATION_PREV | 0 | 921600 (18000) | (seconds) |
| Instant time SUUNTO_TIME_PREV | 0 | 86400 | (sec from midnight) |
| Instant duration SUUNTO_DURATION_PREV | 0 | 921600 (18000) | (seconds) |
| Fastest distance SUUNTO_FASTEST_DISTANCE | 0 | 65535 (18000) | (s/km OR s/mi) |
| Fastest distance unit SUUNTO_FASTEST_DISTANCE_UNIT | 0 | 1 | (0=km, 1=mi) |

## SWIMMING

| | | | |
|---|---|---|---|
| Pool length SUUNTO_SWIMMING_POOL_LENGTH | 0 (25) | 10000 (200) | (m) |
| Strokes SUUNTO_SWIMMING_STROKES | 0 | 2147483647 (100) | (nbr of strokes) |
| Rest time SUUNTO_SWIMMING_REST_TIME | 0 | 2147483647 (720) | (seconds) |
| Previous pool length duration SUUNTO_SWIMMING_PREVIOUS_POOL_LENGTH_DURATION | 0 | 2147483647 (120) | (seconds) |

| | | | |
|---|---|---|---|
| Previous pool length strokes SUUNTO_SWIMMING_PREVIOUS_POOL_LENGTH_STROKES | 0 | 2147483647 (100) | (nbr of strokes) |
| Interval duration SUUNTO_SWIMMING_INTERVAL_DURATION | 0 | 2147483647 (3600) | (seconds) |
| Interval distance SUUNTO_SWIMMING_INTERVAL_DISTANCE | 0 | 2147483647 (1000) | (m) |
| Interval strokes SUUNTO_SWIMMING_INTERVAL_STROKES | 0 | 2147483647 (1000) | (nbr of strokes) |
| Tail avg strokes[30sec] SUUNTO_SWIMMING_STROKES_AVG | 0 | 2147483647 (1000) | (nbr of strokes) |
| Tail max strokes[30sec] SUUNTO_SWIMMING_STROKES_MAX | 0 | 2147483647 (1000) | (nbr of strokes) |
| Tail min strokes[30sec] SUUNTO_SWIMMING_STROKES_MIN | 0 | 2147483647 (1000) | (nbr of strokes) |
| Tail diff strokes[30sec] SUUNTO_SWIMMING_STROKES_DIFF | 0 | 2147483647 (1000) | (nbr of strokes) |
| Previous pool length style SUUNTO_SWIMMING_PREVIOUS_POOL_LENGTH_STYLE | 0 | 5 | (0=other,1=butterfly, 2=backstroke, 3= breaststroke, 4=freestyle, 5 = drill) |

## POWER

| | | | |
|---|---|---|---|
| Bike power SUUNTO_BIKE_POWER | 0 | 9999 (500) | (W) |
| Average bike power SUUNTO_BIKE_AVG_POWER | 0 | 9999 (500) | (W) |
| Maximum bike power SUUNTO_BIKE_MAX_POWER | 0 | 9999 (500) | (W) |
| Lap avg bike power SUUNTO_LAP_BIKE_AVG_POWER | 0 | 9999 (500) | (W) |
| Lap max bike power SUUNTO_LAP_BIKE_MAX_POWER | 0 | 9999 (500) | (W) |
| Tail avg power[30sec] SUUNTO_BIKE_POWER_AVG | 0 | 9999 (500) | (W) |
| Tail max power[30sec] SUUNTO_BIKE_POWER_MAX | 0 | 9999 (500) | (W) |
| Tail min power[30sec] SUUNTO_BIKE_POWER_MIN | 0 | 9999 (500) | (W) |
| Tail diff power[30sec] SUUNTO_BIKE_POWER_DIFF | 0 | 9999 (500) | (W) |
| Bike power connected SUUNTO_BIKE_POWER_CONNECTED | 0 | 1 | (0=not connected, 1=connected) |

## CADENCE

| | | | |
|---|---|---|---|
| Cadence SUUNTO_CADENCE | 0 | 240 (199) | (rpm) |
| Average cadence SUUNTO_AVG_CADENCE | 0 | 240 (199) | (rpm) |
| Maximum cadence SUUNTO_MAX_CADENCE | 0 | 240 (199) | (rpm) |
| Lap avg cadence SUUNTO_LAP_AVG_CADENCE | 0 | 240 (199) | (rpm) |
| Lap max cadence SUUNTO_LAP_AVG_CADENCE_PREV | 0 | 240 (199) | (rpm) |
| Previous lap avg cadence SUUNTO_LAP_MAX_CADENCE | 0 | 240 (199) | (rpm) |
| Previous lap max cadence SUUNTO_LAP_MAX_CADENCE_PREV | 0 | 240 (199) | (rpm) |
| Instant cadence SUUNTO_CADENCE_PREV | 0 | 240 (199) | (rpm) |
| Instant avg cadence SUUNTO_AVG_CADENCE_PREV | 0 | 240 (199) | (rpm) |

| | | | |
|---|---|---|---|
| Tail avg cadence[30sec] SUUNTO_CADENCE_AVG | 0 | 240 (199) | (rpm) |
| Tail max cadence[30sec] SUUNTO_CADENCE_MAX | 0 | 240 (199) | (rpm) |
| Tail min cadence[30sec] SUUNTO_CADENCE_MIN | 0 | 240 (199) | (rpm) |
| Tail diff cadence[30sec] SUUNTO_CADENCE_DIFF | 0 | 240 (199) | (rpm) |

## PERSONAL

| | | | |
|---|---|---|---|
| User max heart rate SUUNTO_USER_MAX_HR | 30 | 240 (229) | (bpm) |
| User rest heart rate SUUNTO_USER_REST_HR | 30 | 240 (229) | (bpm) |
| User age SUUNTO_USER_AGE | 0 (10) | 150 (99) | (year) |
| User weight SUUNTO_USER_WEIGHT | 30 | 200 | (kg) |
| User activity class SUUNTO_USER_ACTIVITY_CLASS | 1 | 10 | |
| User height SUUNTO_USER_HEIGHT | 89 (90) | 241 (240) | (cm) |
| User gender SUUNTO_USER_GENDER | 0 | 1 | (0=female, 1=male) |
| User recovery time SUUNTO_USER_RECOVERY_TIME | 0 | 15360 (120) | (h) |
| Activity type SUUNTO_ACTIVITY_TYPE | 1 | 65535 (82) | (ID, list of activities below) |

Not specified sport = 1
Multisport = 2
Run = 3
Cycling = 4
MountainBiking = 5
Swimming = 6
Skating = 8
Aerobics = 9
YogaPilates = 10
Trekking = 11
Walking = 12
Sailing = 13
Kayaking = 14
Rowing = 15
Climbing = 16
Indoor cycling = 17
Circuit training = 18
Triathlon = 19
Alpine skiing = 20
Snowboarding = 21
Crosscountry skiing = 22
Weight training = 23
Basketball = 24
Soccer = 25
Ice Hockey = 26
Volleyball = 27
Football = 28
Softball = 29
Cheerleading = 30
Baseball = 31
Tennis = 33
Badminton = 34
Table tennis = 35
Racquet ball = 36
Squash = 37
Combat sport = 38
Boxing = 39
Floorball = 40
Scuba diving = 51
Free diving = 52
Adventure Racing = 61
Bowling = 62
Cricket = 63
Cross trainer = 64
Dancing = 65
Golf = 66
Gymnastics = 67
Handball = 68
Horseback riding = 69
Ice Skating = 70
Indoor Rowing = 71
Canoeing = 72
Motorsports = 73
Mountaineering = 74
Orienteering = 75
Rugby = 76
Ski Touring = 78
Stretching = 79
Telemark skiing = 80
Track and Field = 81
Trail Running = 82
Open water swimming = 83
Nordic walking = 84
Snow shoeing = 85
Windsurfing/Surfing = 86
Kettlebell = 87
Roller skiing = 88
Standup paddling (SUP) = 89
Cross fit = 90
Kitesurfing/Kiting = 91
Paragliding = 92
Treadmill = 93
Frisbee = 94
Indoor training = 95

# NOT SUPPORTED FEATURES

Some common software engineering features are not supported. Here are few examples.

- Arrays/vectors
- Switch statement
- Loops (Actually the App is always in a loop because it’s ran once per second.)
- Objects
- mean(x), min(x),max(x), std(x)