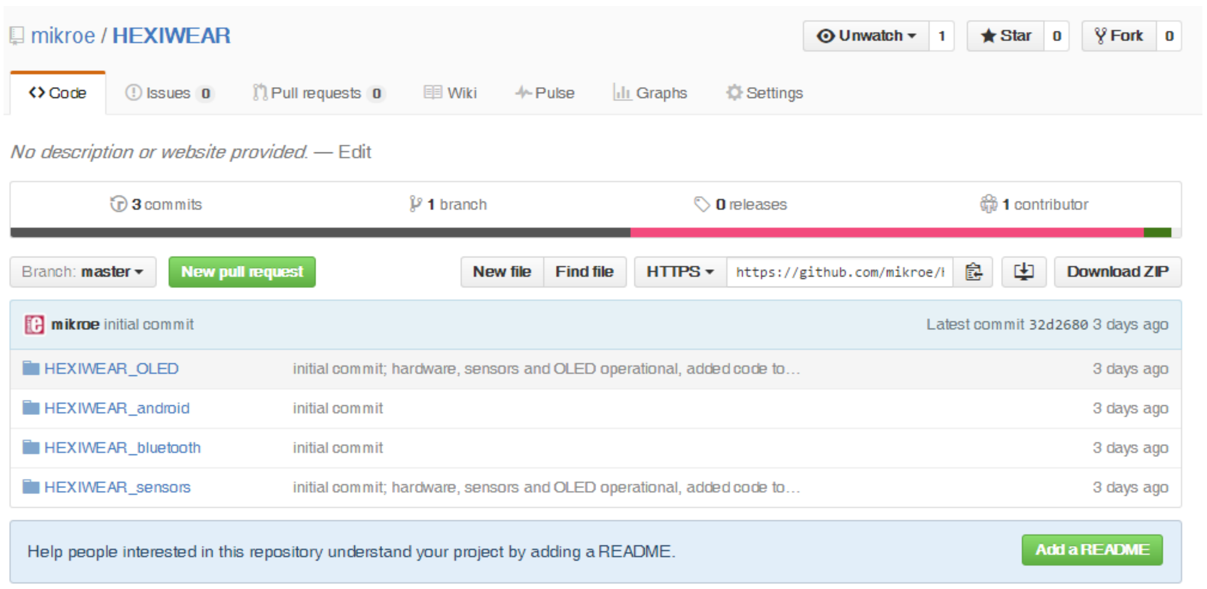
HEXIWEAR

Getting Started

Currently, there are two projects for Kinetis Design Studio 3.0, uploaded at Github, which are to be used for exploring the hardware functionality of HEXIWEAR smart watch.

Github's HEXIWEAR repository address is here.


mikroe / HEXIWEAR

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

No description or website provided. — Edit

3 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | New file | Find file | HTTPS | https://github.com/mikroe/H | Download ZIP

mikroe initial commit — Latest commit 32d2680 3 days ago

HEXIWEAR_OLED	initial commit; hardware, sensors and OLED operational, added code to…	3 days ago
HEXIWEAR_android	initial commit	3 days ago
HEXIWEAR_bluetooth	initial commit	3 days ago
HEXIWEAR_sensors	initial commit; hardware, sensors and OLED operational, added code to…	3 days ago

Help people interested in this repository understand your project by adding a README. — Add a README

Project HEXIWEAR_sensors

This project is aimed for exploring the sensors' communication and functionality.

HEXIWEAR hardware is comprised of the following sensors:

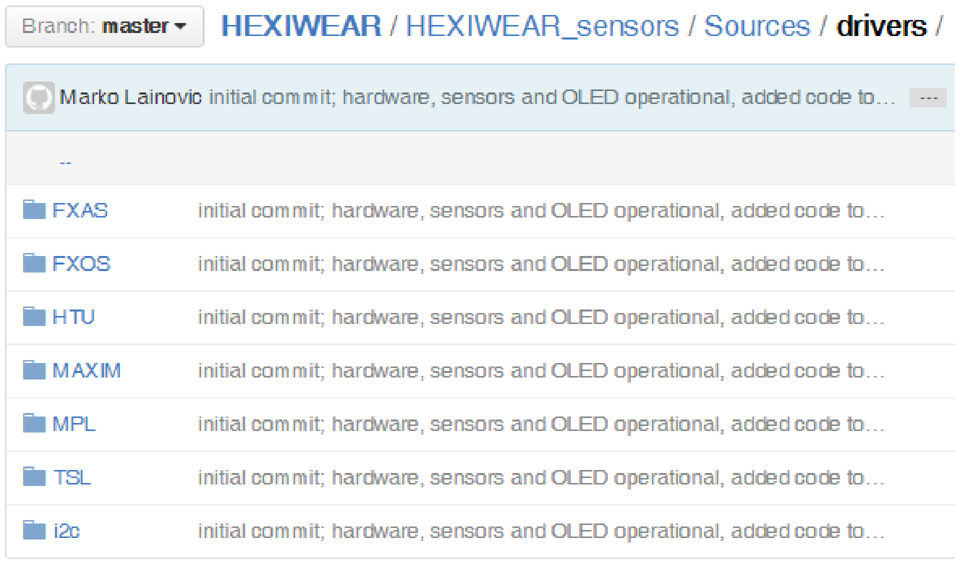
- Freescale **FXAS21002C** 3-axis digital gyroscope
- Freescale **FXOS8700CQ** 3-axis digital accelerometer and 3-axis digital magnetometer
- Freescale **MPL3115A2** Precision Altimeter
- TAOS **TSL2561** light-to-digital converter
- MEAS **HTU21D** digital humidity and temperature sensor
- MAXIM **MAX30101** heart-rate Sensor

All sensors are communicated via I2C protocol, each with its unique address. Freescale sensors are always turned on and are connected to module I2C1 bus, while other non-Freescale sensors have a power supply which can be turned on and off using PTB13 pin and are also using separate I2C0 bus. Heart rate sensor is having the additional power supply which can be controlled with PTA29 pin. These power supplies' states have to be accounted for in order to use these sensors.

Sensors return data which is broadly classified into three categories:

- Motion data (accelerometer, magnetometer and gyroscope data)
- Weather data (humidity, pressure and temperature data)
- Health data (heart rate data)

Drivers for the sensors can be found under the "Sources/drivers/" folder.


Branch: master
HEXIWEAR / HEXIWEAR_sensors / Sources / drivers /

Marko Lainovic initial commit; hardware, sensors and OLED operational, added code to...

..

FXAS	initial commit; hardware, sensors and OLED operational, added code to...
FXOS	initial commit; hardware, sensors and OLED operational, added code to...
HTU	initial commit; hardware, sensors and OLED operational, added code to...
MAXIM	initial commit; hardware, sensors and OLED operational, added code to...
MPL	initial commit; hardware, sensors and OLED operational, added code to...
TSL	initial commit; hardware, sensors and OLED operational, added code to...
i2c	initial commit; hardware, sensors and OLED operational, added code to...

The general structure for all the drivers is the same: source files implementing the driver functionality have a "`_driver`" suffix, while the structures containing sensor information and user settings are located in source files with a "`_defs`" suffix. Constants and other relevant information are in header files with "`_info`" suffix. Relevant structure types and enumerations are located in header files with suffix "`_types`".

Sensors' drivers are generally relying on KSDK drivers and wrappers located in their respective folders, e.g. the sensors in this project are either using KSDK `fsl_i2c_master_driver` functions directly or are using I2C wrappers located in the "Sources/drivers/I2C/" folder.

The sensor utilizing code is located in the "Sources/sensors" folder, which is utilizing the drivers above. The sensors' functionality is embedded into a FreeRTOS task `sensor_GetData()`. The sensors are initialized via function `sensor_InitModules()`. Afterwards, the task is fetching the sensor's data subsequently in an infinite loop. All the sensors' data is firstly fetched in its default raw data form and then formatted into a desired format before sending the data-packet. Currently, the end data type for all the sensor's data is chosen to be `mE_t` type; `mE_t` type is `uint16_t` type with data converted to a decimal form with 2 decimal places and multiplied by 100, e.g. if the sensor value is 101,35 kPa, then sensor data is saved as `uint16_t` value 10135.

After getting and formatting data, it's packed afterwards into a data-packet, which type is defined in the `host_mcu_interface.h` header file located at "Sources/intf/inc" relative address. These data-packets are exchanged via UART between the host MCU MK64 and the

bluetooth module KW40Z, with basic packet confirmation.
Tasks defined in `host_mcu_interface_rx.c` and `host_mcu_interface_tx.c` files are in charge of sending and receiving data-packets.
All the packets are sent via an UART wrapper `HostInterface_FlushPacket()` waiting on a queue to be filled with packet data. This function is used only by one task at a time (using mutex mechanism).
Intern function `packetHandler()`, which represents the state machine for analyzing data-packets, is implemented as a function waiting on a queue to be filled with a data-packet within an UART RX callback function.

Project HEXIWEAR_OLED

This project is utilizing OLED screen and FLASH functionality. These modules are using SPI bus to communicate with host MCU. Similarly, the drivers functions either directly use fsl_dspi_master functions or use SPI wrappers located at "drivers/SPI" address.
POWER drivers are intended for entering into various sleep modes and for turning on/off various modules in the system.

OLED drivers implement the drawing of basic objects and images, transition effects, and other OLED screen characteristics which can be controlled via OLED commands defined in `OLED_info.h` header file. OLED screen also has a dedicated pin PTC13 for turning on/off the 15V power supply.

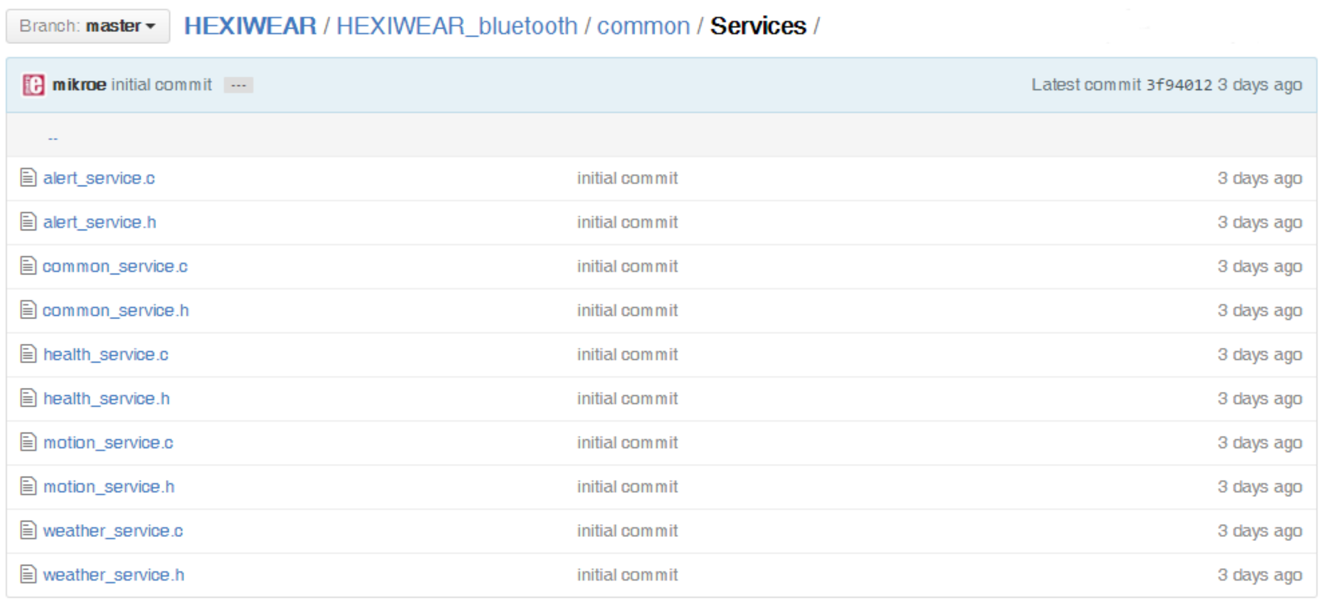
Changing screens should always be performed using menu driver functions. Menu drivers are located in the menu folder, and use OLED driver functions for drawing images and objects. Images are located in the `menu_resources.c` file, and screen objects pertaining to those images are defined in the `menu_screens.c` file. Screen objects are organized as structures containing references to adjacent screens and to the screens up and below in the screen hierarchy, as well as pointers to an optional screen initialization function and a callback.

Project HEXIWEAR_bluetooth

This section describes implementation of firmware for Bluetooth LE SoC Kinetis KW40Z. IAR Embedded Workbench version 7.40 is required; firmware is compiled and tested with version 7.40.3.

The firmware works as a GAP peripheral node, by firstly entering the GAP Discoverable mode and then it waits for the GAP central node to connect.

Drivers for implemented services can be found under the "common/Service" folder. For the information about currently implemented services, refer to the "**HEXIWEAR BLUETOOTH SPECIFICATIONS**" document.

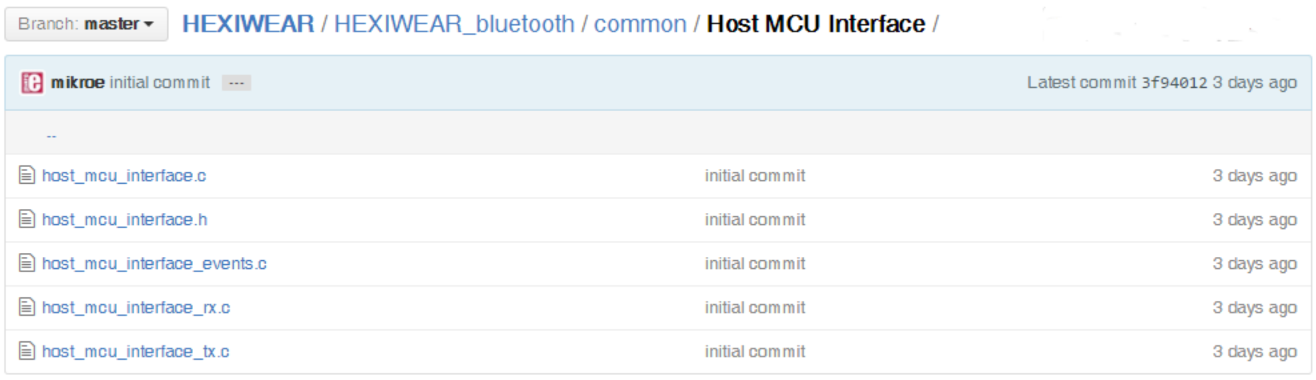


Branch: master ▾ HEXIWEAR / HEXIWEAR_bluetooth / common / **Services** /

mikroe initial commit ··· Latest commit 3f94012 3 days ago

..		
alert_service.c	initial commit	3 days ago
alert_service.h	initial commit	3 days ago
common_service.c	initial commit	3 days ago
common_service.h	initial commit	3 days ago
health_service.c	initial commit	3 days ago
health_service.h	initial commit	3 days ago
motion_service.c	initial commit	3 days ago
motion_service.h	initial commit	3 days ago
weather_service.c	initial commit	3 days ago
weather_service.h	initial commit	3 days ago

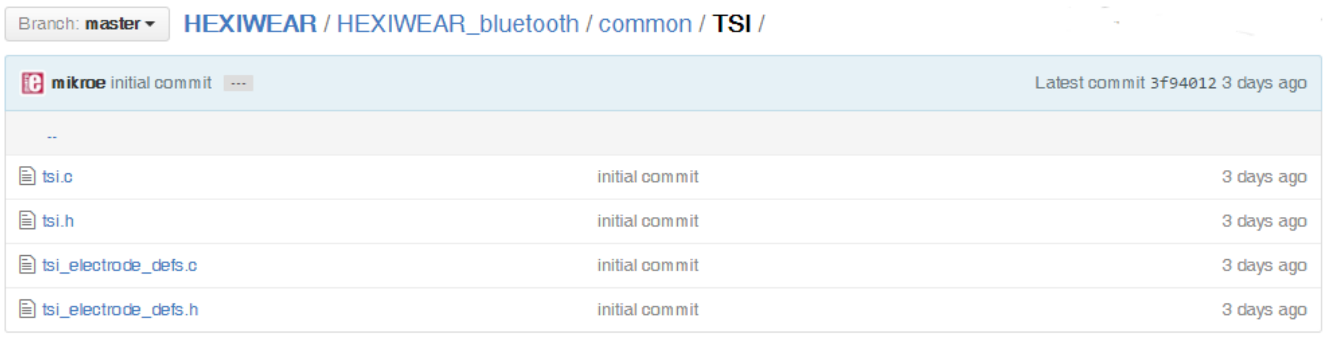
KW40Z chip communicates with host MCU MK64 through UART interface, throught the exchange of AT packets. Each received packet needs to be confirmed and for every sent packet, driver will wait for the confirmation on the host side. Drivers for the communication with the host MCU can be found in the "common/Host MCU Interface" folder.


Branch: master ▾ HEXIWEAR / HEXIWEAR_bluetooth / common / Host MCU Interface /

mikroe initial commit … Latest commit 3f94012 3 days ago

..		
host_mcu_interface.c	initial commit	3 days ago
host_mcu_interface.h	initial commit	3 days ago
host_mcu_interface_events.c	initial commit	3 days ago
host_mcu_interface_rx.c	initial commit	3 days ago
host_mcu_interface_tx.c	initial commit	3 days ago

Drivers for handling the state of capacitive electrodes can be found in the "common/TSI" folder. Driver implements handling of the "on-down" and "press" events for all the electrodes and of the "slide" event for "vertical pair of electrodes".


Branch: master ▾ HEXIWEAR / HEXIWEAR_bluetooth / common / TSI /

mikroe initial commit … Latest commit 3f94012 3 days ago

..		
tsi.c	initial commit	3 days ago
tsi.h	initial commit	3 days ago
tsi_electrode_defs.c	initial commit	3 days ago
tsi_electrode_defs.h	initial commit	3 days ago

Project HEXIWEAR_android

The application demonstrates reading and displaying basic data from the HEXIWEAR module, as well as sending the information on notifications received from a phone (SMS, e-mail, incoming call..). Application is tested on a Nexus 5 device.

After the application starts, it will enter scan mode and search for the HEXIWEAR module by its address. Upon successful search, connection and service discovery procedures are started automatically.

Disclaimer: please note that the firmware is at an early stage of development and not all functionality is released.