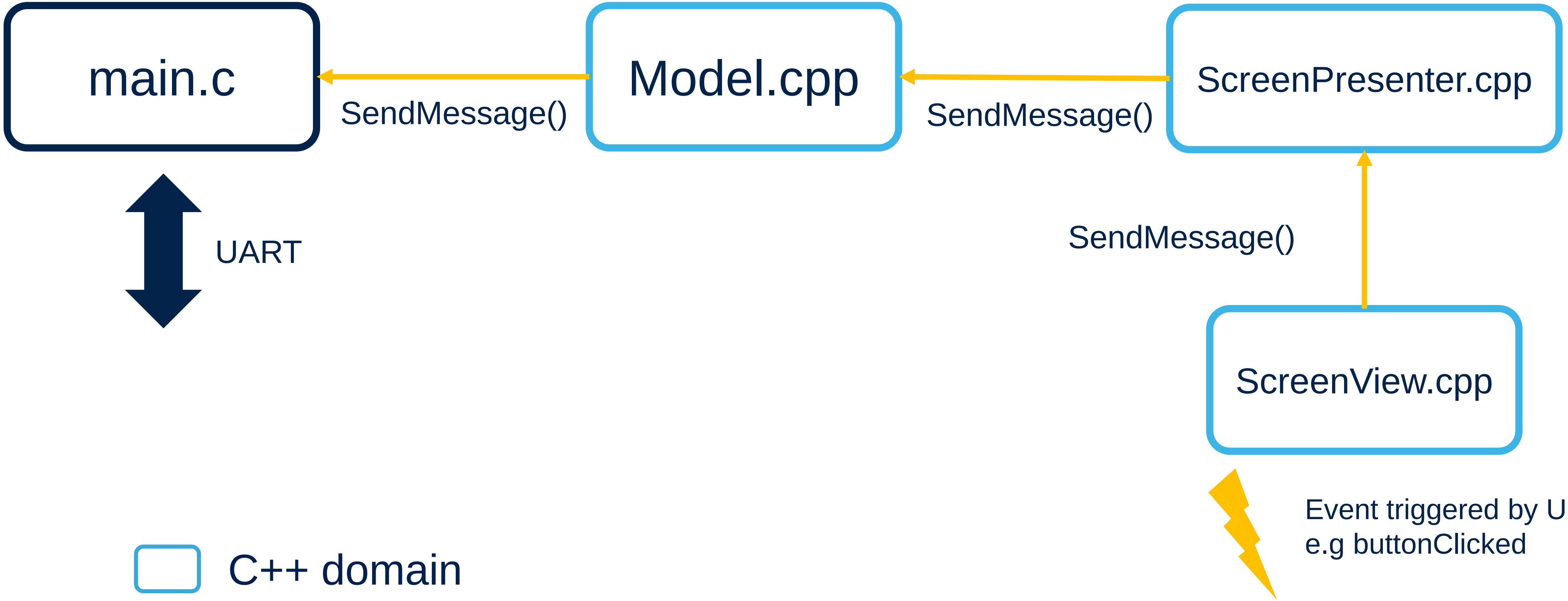
UI to Backend – UART example (no OS)

life.augmented

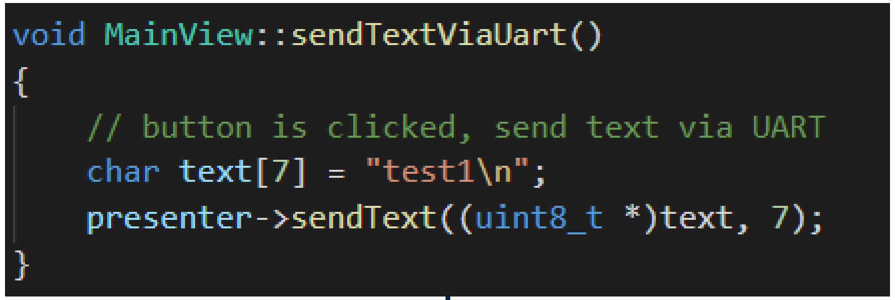
UART example – Overview - UI to Backend



▢ C++ domain

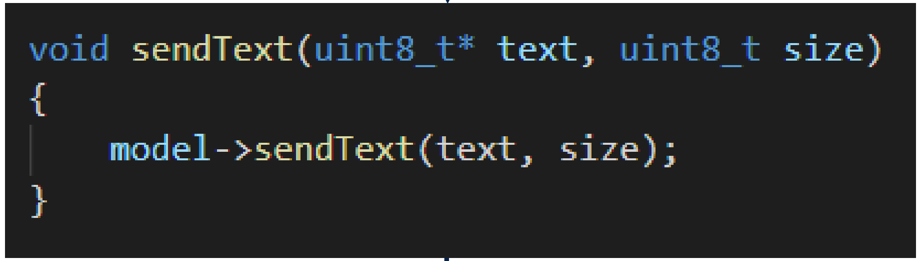
▢ C domain

UART example – Overview - UI to Backend - Code



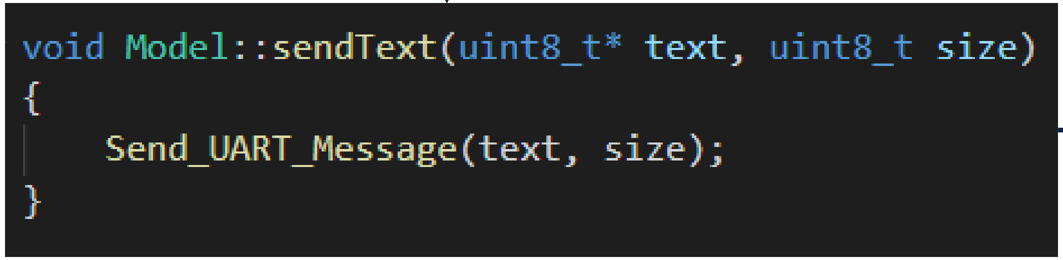
```
void MainView::sendTextViaUart()
{
    // button is clicked, send text via UART
    char text[7] = "test1\n";
    presenter->sendText((uint8_t *)text, 7);
}
```

MainView.cpp



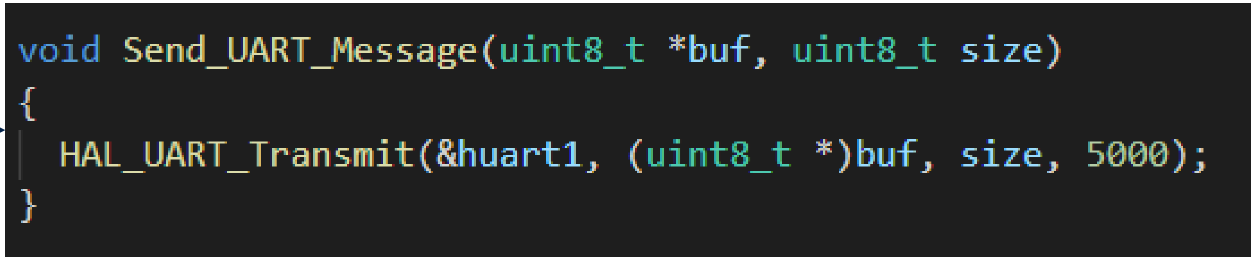
```
void sendText(uint8_t* text, uint8_t size)
{
    model->sendText(text, size);
}
```

MainPresenter.hpp



```
void Model::sendText(uint8_t* text, uint8_t size)
{
    Send_UART_Message(text, size);
}
```

Model.cpp



```
void Send_UART_Message(uint8_t *buf, uint8_t size)
{
  HAL_UART_Transmit(&huart1, (uint8_t *)buf, size, 5000);
}
```

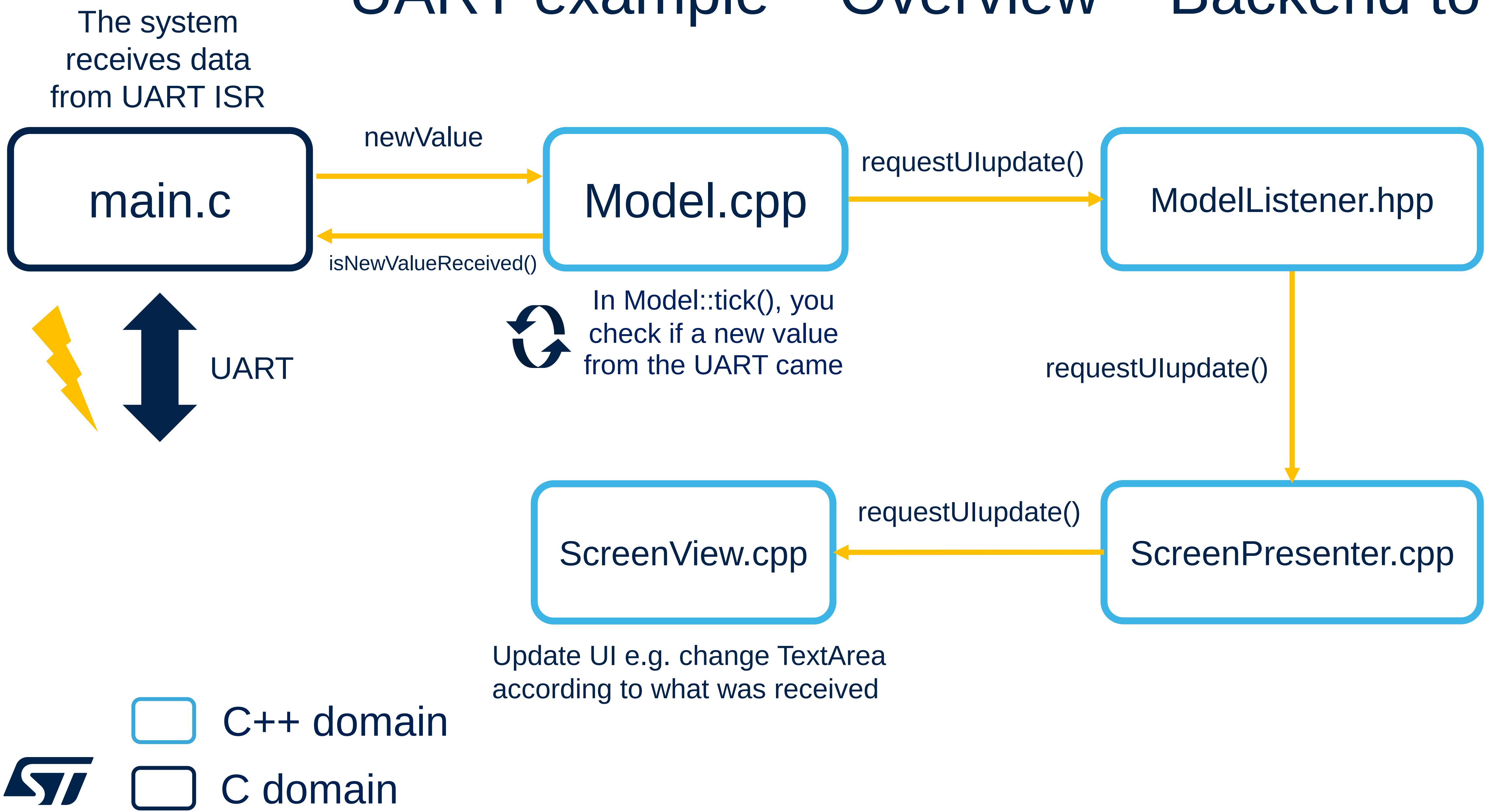
main.c

life.augmented

Backend to UI (no OS)

life.augmented

UART example – Overview – Backend to UI



The system receives data from UART ISR

main.c

newValue

isNewValueReceived()

Model.cpp

requestUIupdate()

ModelListener.hpp

UART

In Model::tick(), you check if a new value from the UART came

requestUIupdate()

ScreenView.cpp

requestUIupdate()

ScreenPresenter.cpp

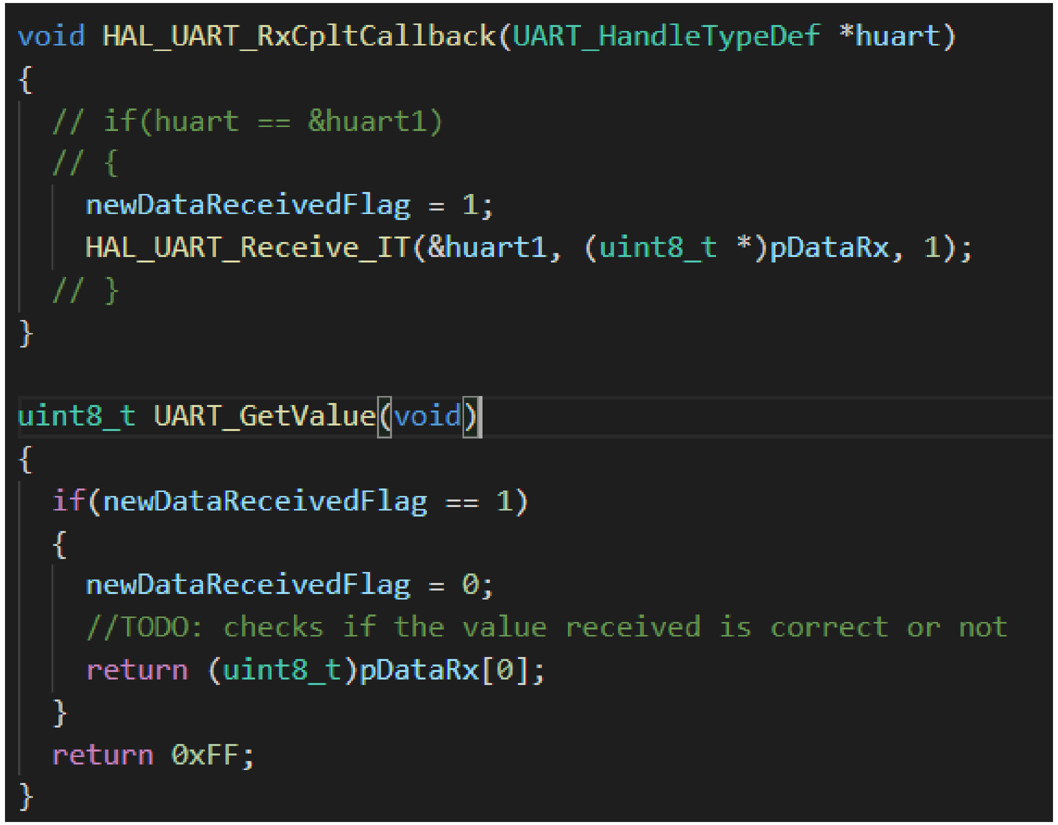
Update UI e.g. change TextArea according to what was received

□ C++ domain

□ C domain

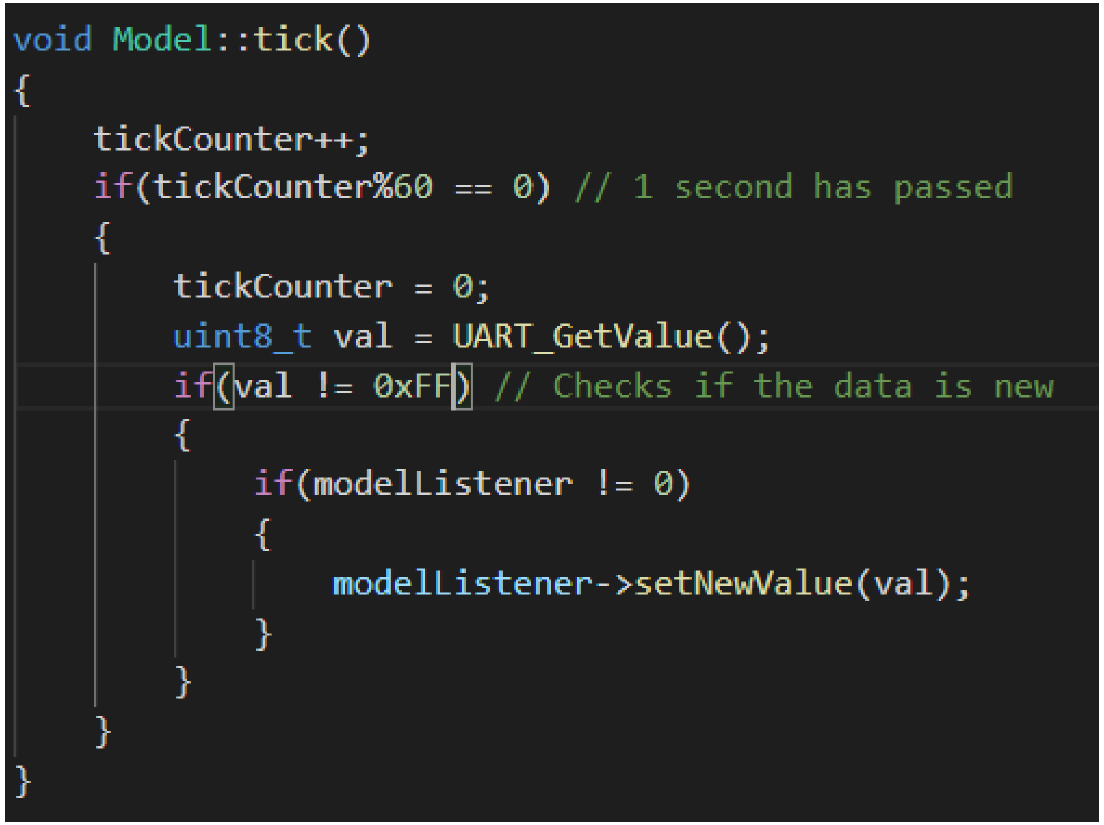
UART example – Overview – Backend to UI – Code (1/2)

main.c



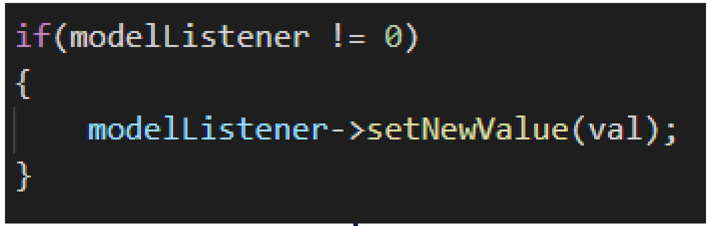
```c
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  // if(huart == &huart1)
  // {
    newDataReceivedFlag = 1;
    HAL_UART_Receive_IT(&huart1, (uint8_t *)pDataRx, 1);
  // }
}

uint8_t UART_GetValue(void)
{
  if(newDataReceivedFlag == 1)
  {
    newDataReceivedFlag = 0;
    //TODO: checks if the value received is correct or not
    return (uint8_t)pDataRx[0];
  }
  return 0xFF;
}
```

Model.cpp



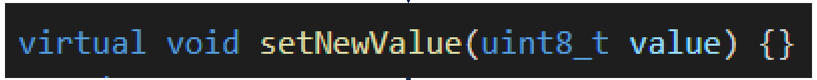
```cpp
void Model::tick()
{
    tickCounter++;
    if(tickCounter%60 == 0) // 1 second has passed
    {
        tickCounter = 0;
        uint8_t val = UART_GetValue();
        if(val != 0xFF) // Checks if the data is new
        {
            if(modelListener != 0)
            {
                modelListener->setNewValue(val);
            }
        }
    }
}
```

UART example – Overview – Backend to UI – Code (2/2)



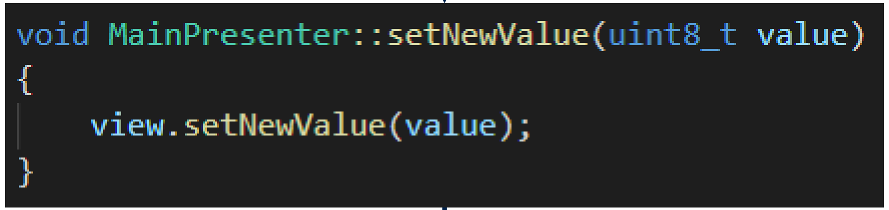
```
if(modelListener != 0)
{
    modelListener->setNewValue(val);
}
```

Model.cpp



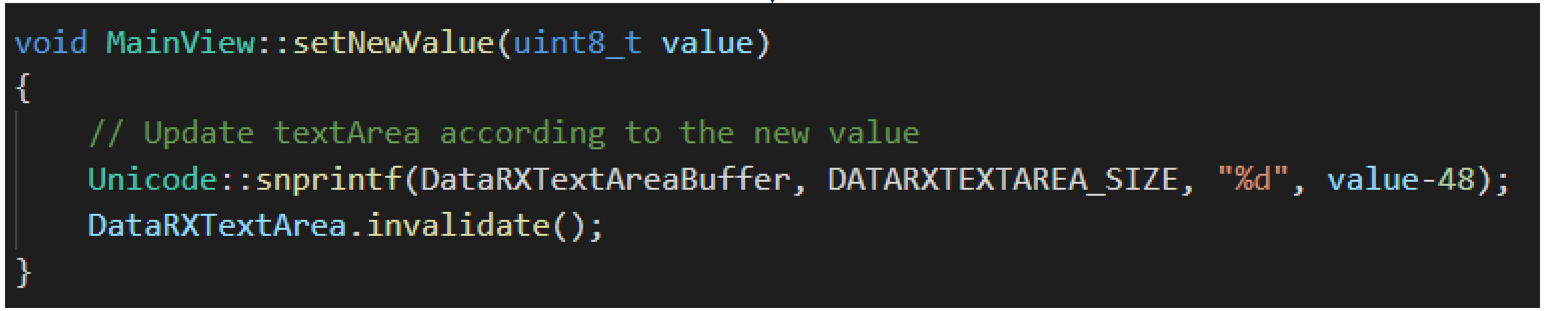
```
virtual void setNewValue(uint8_t value) {}
```

ModelListener.hpp



```
void MainPresenter::setNewValue(uint8_t value)
{
    view.setNewValue(value);
}
```

MainPresenter.cpp

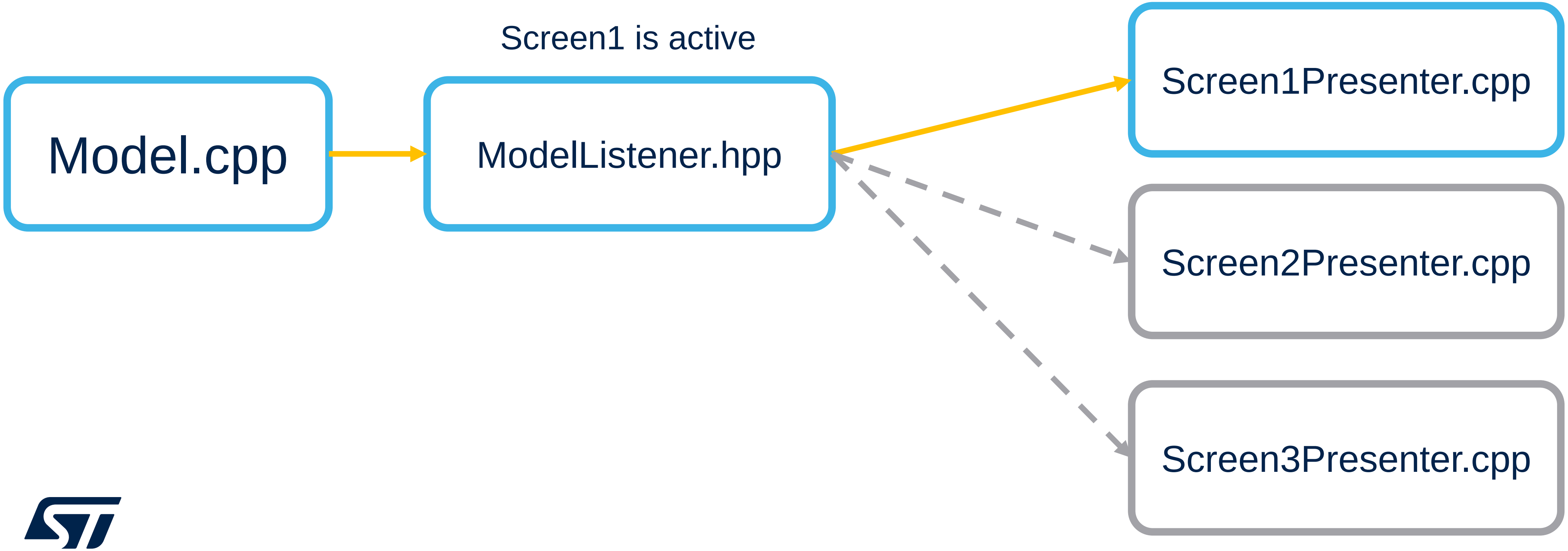


```
void MainView::setNewValue(uint8_t value)
{
    // Update textArea according to the new value
    Unicode::snprintf(DataRXTextAreaBuffer, DATARXTEXTAREA_SIZE, "%d", value-48);
    DataRXTextArea.invalidate();
}
```

MainView.cpp

What’s the ModelListener ?

- The Model has a pointer to the currently active Presenter. The type of this pointer is an interface (ModelListener) which you can modify to reflect the application-specific events that are appropriate.



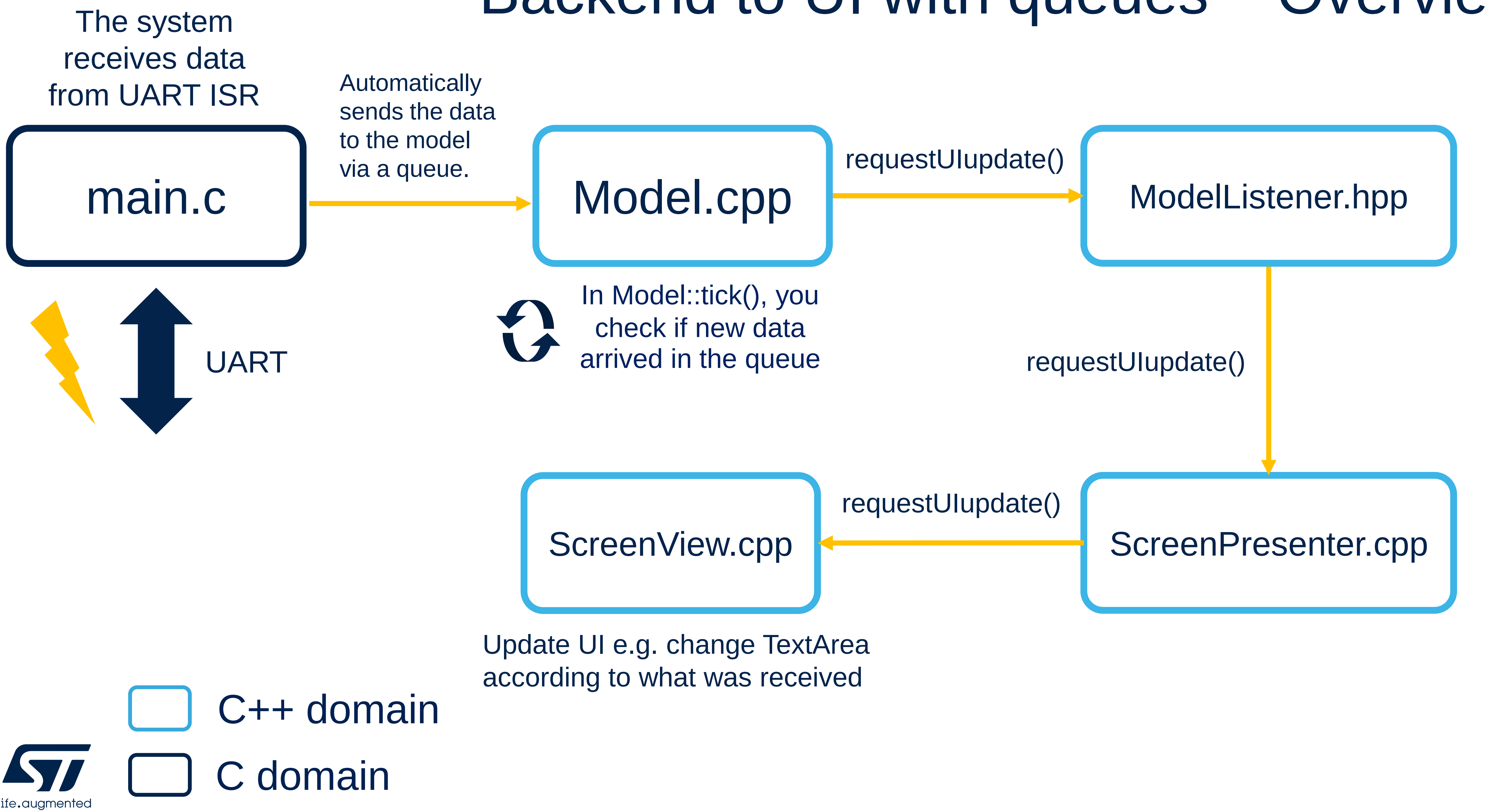
Backend to UI (with OS)

life.augmented

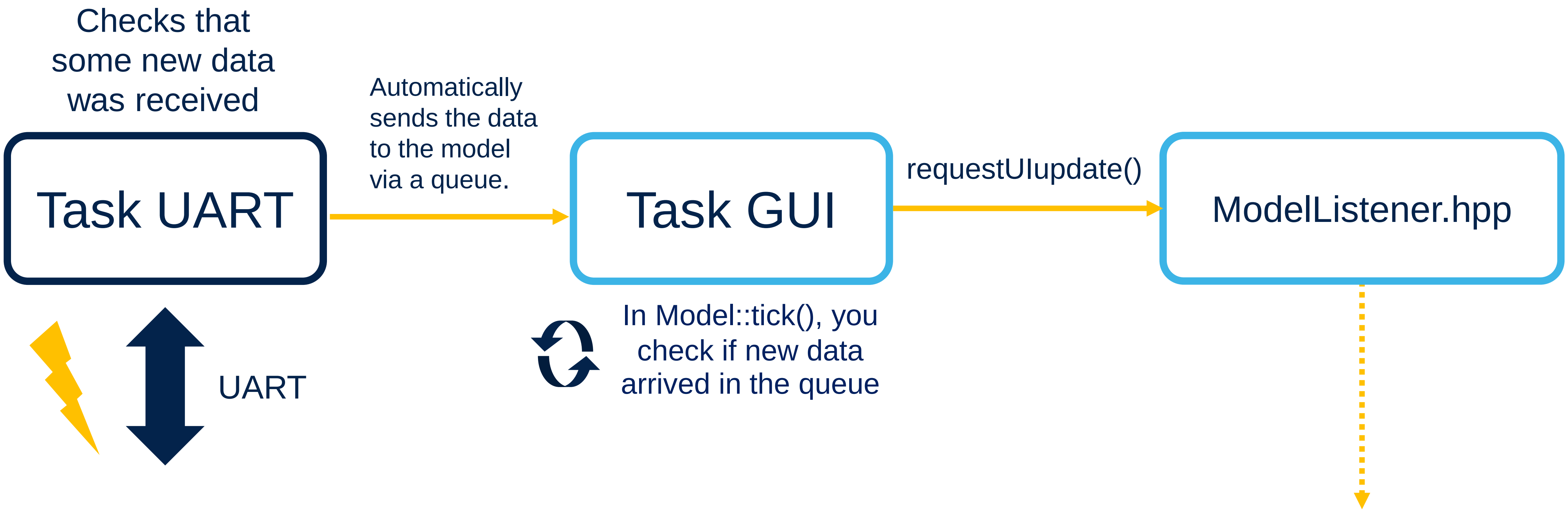
When using FreeRTOS - Queues

- Prerequisite : None. No need to learn extensively how FreeRTOS works.
- When using FreeRTOS, or any Embedded OS, you most likely use different tasks.
- To send information from one task to the other, you need something called a **queue**.
- Queues have 2 main benefits :
  - Provide a way to communicate between tasks.
  - A non-blocking communication system.

Backend to UI with queues – Overview


The system receives data from UART ISR
main.c
UART
Automatically sends the data to the model via a queue.
Model.cpp
In Model::tick(), you check if new data arrived in the queue
requestUIupdate()
ModelListener.hpp
requestUIupdate()
ScreenPresenter.cpp
requestUIupdate()
ScreenView.cpp
Update UI e.g. change TextArea according to what was received
C++ domain
C domain

Queues for multi-tasks communication



Checks that some new data was received

Task UART

Automatically sends the data to the model via a queue.

Task GUI

requestUIupdate()

ModelListener.hpp

UART

In Model::tick(), you check if new data arrived in the queue

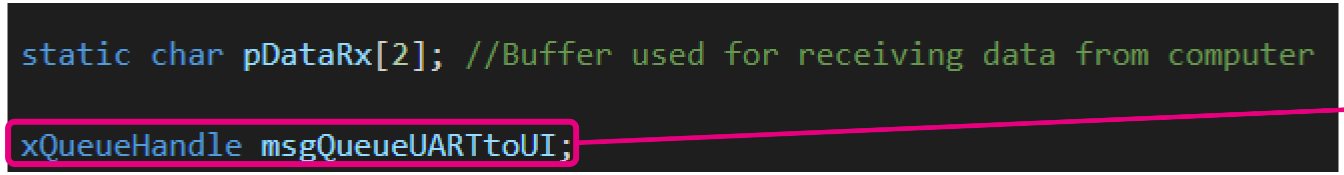
□ C++ domain

□ C domain

FreeRTOS Queue API

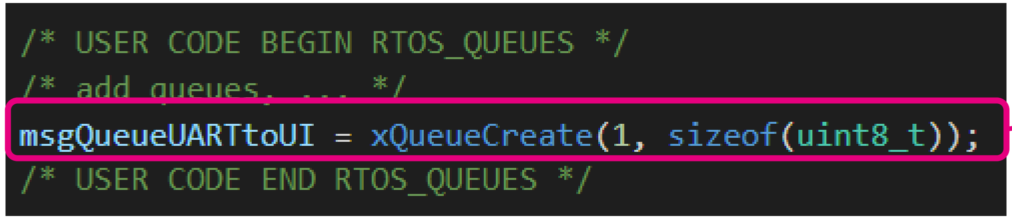
- For using queues with FreeRTOS you only need to know the following elements.
  - A queue is declared like this :
    - xQueueHandle myQueue;
  - A queue is created as follows :
    - myQueue = xQueueCreate(nbElements, sizeof(element));
  - To add an element in a queue :
    - xQueueSendFromISR(myQueue, &element, 0); // When call inside an interrupt handler
    - xQueueSendToBack(myQueue, &element, 0); //When called from a task
  - To check if an element is in the queue :
    - if (uxQueueMessagesWaiting(myQueue) > 0) { /* Retrieve new data */ }
  - To take the element from the queue :
    - xQueueReceive(myQueue, &newValue, 0); // newValue is the new value received from the queue

Backend to UI with queues – Example – main.c



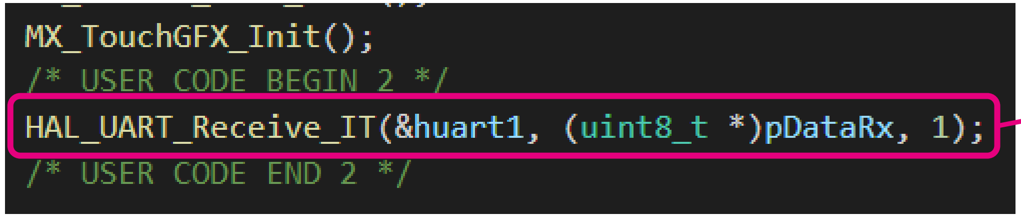
```c
static char pDataRx[2]; //Buffer used for receiving data from computer

xQueueHandle msgQueueUARTtoUI;
```

Declaration of the queue



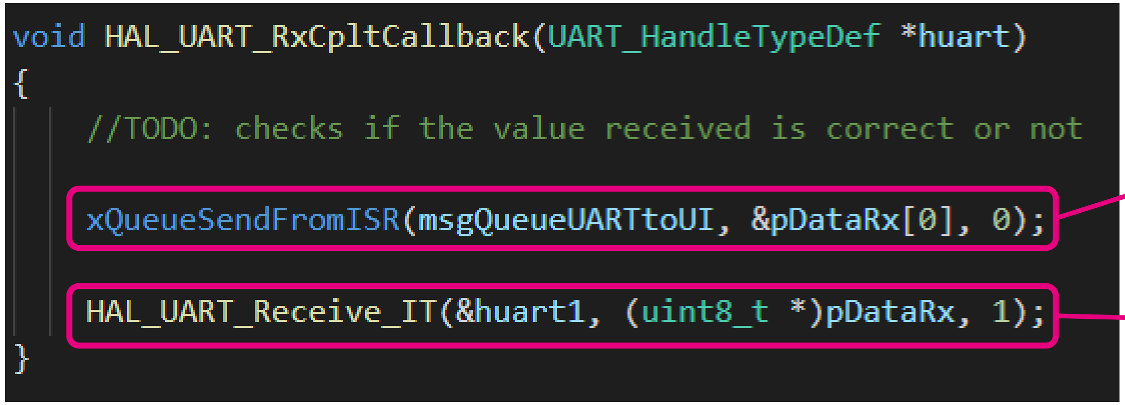
```c
/* USER CODE BEGIN RTOS_QUEUES */
/* add queues, ... */
msgQueueUARTtoUI = xQueueCreate(1, sizeof(uint8_t));
/* USER CODE END RTOS_QUEUES */
```

Creation of a queue of size one uint8_t element



```c
MX_TouchGFX_Init();
/* USER CODE BEGIN 2 */
HAL_UART_Receive_IT(&huart1, (uint8_t *)pDataRx, 1);
/* USER CODE END 2 */
```

Ready to receive a value through UART

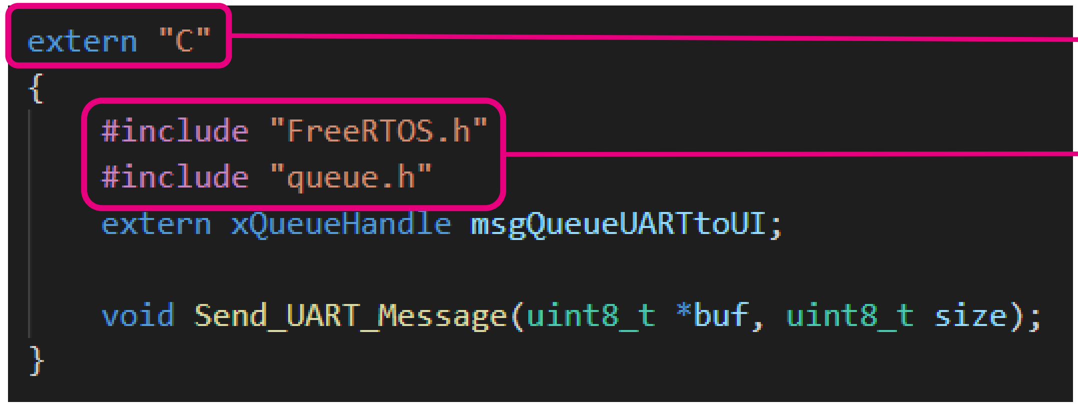


```c
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    //TODO: checks if the value received is correct or not

    xQueueSendFromISR(msgQueueUARTtoUI, &pDataRx[0], 0);

    HAL_UART_Receive_IT(&huart1, (uint8_t *)pDataRx, 1);
}
```

Send data received through UART to the queue

Ready to receive a new value through UART

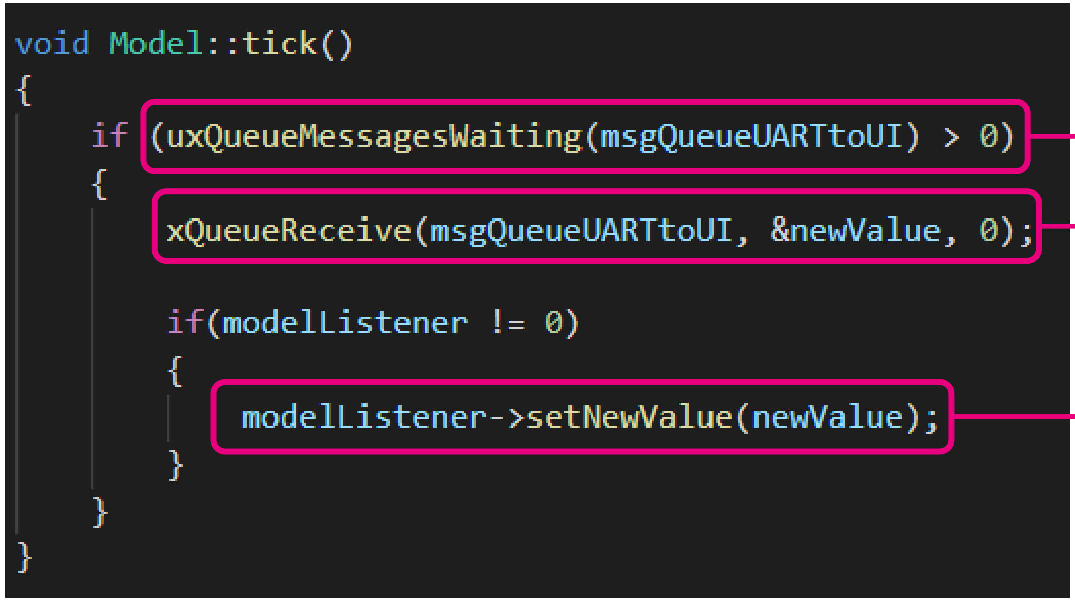
Backend to UI with queues – Example – Model.cpp



```
extern "C"
{
    #include "FreeRTOS.h"
    #include "queue.h"
    extern xQueueHandle msgQueueUARTtoUI;

    void Send_UART_Message(uint8_t *buf, uint8_t size);
}
```

We are in a C++ file, so everything linked to a C file needs to be encapsulated in extern “C”

Necessary includes to be able to use queues



```
void Model::tick()
{
    if (uxQueueMessagesWaiting(msgQueueUARTtoUI) > 0)
    {
        xQueueReceive(msgQueueUARTtoUI, &newValue, 0);

        if(modelListener != 0)
        {
            modelListener->setNewValue(newValue);
        }
    }
}
```

Checking if a new value is in the queue

Receiving the new value from the queue

Update UI according to the new value